

Nº8 – Julio 2004

analisis

SUPER CARS

A WHOLE NEW BALL GAME

al descubierto

LA SAGA WALLY, PATAS ARRIBA

CREANDO UNA AVENTURA CONVERSACIONAL CON Z88DK (II)

zona www

ESPECTRUM

1er aniversario

HARRY
(Everyone's a Wally)

...y mucho más

www.speccy.org/magazinezx





Redacción:

Santiago Romero (SROMERO).
Federico Álvarez (FALVAREZ).
Pablo Suau (SIEW).
Miguel A. García Prada (DEVIL_NET).

Ilustración de Portada:

Juanje Gómez (DEV).

Colaboraciones en este número:

Jaime Tejedor (METALBRAIN).
S.T.A.R.

Maquetación en PDF

Álvaro Alea (ALEASOFT)

Contacto:

magazine@speccy.org



Redacción de MAGAZINE ZX

Hace algo más de un año que tres personas empezamos a darle vueltas a la cabeza sobre cómo hacer una publicación con una frecuencia más o menos estable, que hablara de todo lo que sucediera en el mundillo que rodea al Spectrum, especialmente en España, dando una opinión crítica, escapando de la adulación fácil, y que pudiera ser accesible para todo el que quisiera tanto leerla como colaborar.

Después de un período de planificación, en el que se barajaron varias posibilidades, en julio de 2003 nació el primer número de Magazine ZX tal como la conocéis. Durante los tres primeros números mantuvimos una periodicidad mensual, con bastante esfuerzo por parte nuestra y de los colaboradores que se iban incorporando. Al finalizar el tercer número decidimos pasar a una frecuencia bimestral, lo que nos permitía pulir más los contenidos y no quemarnos excesivamente pronto obligándonos a escribir en plazos demasiado cortos.

Durante este año que llevamos ofreciendo esta publicación centrada en el Spectrum con contenidos de todo tipo, análisis de páginas web, juegos, montajes de hardware, cursos de programación, artículos de opinión, noticias, etcétera, hemos recibido halagos y críticas. Ambas cosas no nos influyen a la hora de trabajar en algo que nos gusta, sin pensar en lo que puedan opinar terceras personas, ya que hacemos una labor que nos agrada, nos apetece y lo elaboramos por nosotros mismos. El día que escribamos teniendo en cuenta lo que que va a pensar uno o si se va a enfadar otro, quizá sea el momento de cerrar.

Estos doce meses, con siete números publicados, y el que tenéis en la pantalla de vuestro ordenador, que hace el octavo, ha sido largos y han sucedido muchas cosas, positivas y negativas, en la escena Spectrum. Desde el gran trabajo de Manuel Gómez con las revistas Microhobby (impagable su labor), la salida de juegos nuevos como Flash Beer y TV Games por parte de WSS, pasando por otros que, por desgracia, dan la sensación de que pueden quedar en papel mojado, ya sea por falta de avances o errores de planteamiento (Castlevania ZX, ZX Fútbol, el entorno de programación WinZX)... Peleas, críticas, felicitaciones. En fin, lo de siempre en un grupo de personas.

Los contenidos de este mes esperamos que sean igual de frescos que los del primer número que pusimos a vuestra disposición. Podéis comenzar con una lectura de nuestra sección de actualidad (Panorama), un repaso a lo acontecido en los últimos dos meses en este mundillo. Continuar con el análisis de esos juegos quizás poco conocidos aunque divertidos y dignos de ser jugados, en este caso Super Cars y Whole New Ball Game. Para los manitas la sección Hardware; en esta entrega conectaremos joysticks de norma Atari a nuestros +2/+3, y sacaremos partido al buen monitor que es el Amstrad CTM644. En la sección Al descubierto, una disección de lo que fue y supuso la saga de Wally para las videoaventuras y su evolución.

S.T.A.R. colabora por primera vez en nuestra publicación (y esperemos que sea la primera de una larga lista) con un artículo de opinión, en su estilo habitual, directo y peculiar, sobre el coloreado de los juegos en Spectrum. SIEW sigue con sus cursos de programación, cada vez más interesantes, y un análisis de una de las páginas más veteranas e interesantes: Espectrum, la web de Horace. Un *slideshow*, gracias a METALBRAIN, con una serie de imágenes de los fans de Spectrum fotografiados con las camisetas Sinclair que hizo Badaman. Y, como casi siempre, la entrevista de HORACE, esta vez a Jose María Pérez Rosado y su hermano Antonio, creadores de Dea Tenebraum.

Por descontado, esperamos que los contenidos de este número os agraden y sean de utilidad. Muchas gracias por vuestra fidelidad y esperamos que disfrutéis mucho más tiempo con nuestra publicación.



Redacción de MAGAZINE ZX

Vamos a comentar los hechos más destacables en el mundillo que rodea al Spectrum que, desde el último número de este Magazine, han hecho temblar las ULAs de nuestros queridos ordenadores, para bien y para mal. Dos meses intensos con algunas situaciones inexplicables, algunas irrisorias y otras que llenan de esperanza.

La revista Microhobby online al 100% y el DVD "oficial" (o una de cal y otra de arena)

De lo mejorcito que pasó en estos meses, la finalización del proyecto Microhobby por parte de Manuel Gómez. Si bien las revistas estaban escaneadas desde hace bastante tiempo, no ha sido hasta hace pocas semanas cuando las hemos podido disfrutar todas online. Este titánico trabajo se vió recompensado con la autorización por parte de Hobby Press para su publicación en internet, ya comentada y debatida en una entrega anterior de Magazine ZX.

Para completar el proyecto, alojado en microhobby.org, varías personas y sitios web, entre los que están El Trastero, SPA2 o Sinclairmania, cedieron diferente material anexo a las revistas para que quedara alojado todo junto: Cintas en formato TZX, catálogos, el Discoflex, etc.

A raiz de terminar de colgar las revistas, se nos hizo saber que se ponía a la venta un DVD con el contenido de microhobby.org al completo por el precio de 10 euros. En este precio se incluía un DVD presentado en su caja correspondiente, con carátula, serigrafía y, en caso de que llegara defectuoso, la posibilidad de cambiarlo por otro en perfectas condiciones. Y empezaron a desfilar ante nuestros ojos situaciones, cuanto menos, un tanto extrañas.

En primer lugar Manuel escribió en el grupo de news ECSS comunicando de antemano su desvinculación con la venta de este DVD, cosa que resultaba un poco chocante. En dicho escrito explicaba, de manera concisa y sin dejar lugar a dudas, que él no había hecho el trabajo esperando una recompensa ni con el fin de conseguirla o lucrarse. A muchos no nos hacía falta esa aclaración para darlo por sentado. Todo iba bien, la venta del DVD que desde la web del proyecto daba a entender o aparentaba estar bajo el auspicio oficial de Hobby Press, y gestionada por José M. Matas (Stalvs), tenía fecha de salida y, suponemos, muchos pedidos.

Poco tiempo después surgió un proyecto alternativo que ofrecía los mismos contenidos pero en esta ocasión en dos DVDs, uno más que en la oferta "oficial" y con presentación más austera: sin caja ni carátula, pero a un precio mucho más reducido de 2,40 euros. Esta iniciativa fue lanzada por Pedrete, explicando que las revistas estaban escaneadas a 200ppp en lugar de a 150ppp, como las ofrecidas en el proyecto "oficial", y que ésta era la causa de tener que emplear un par de DVDs.

Cuando nos congratulábamos de poder tener opciones para elegir donde hacernos con el material, se desata otra de las ya habituales broncas sin sentido en el mundillo. Stalvs acusa a Pedrete de vender el DVD para fastidiarle, sin más, y argumenta que el único que tiene autorización para distribuirlos es él. Justifica el precio por el empleo de grabadoras que no posee y una lista extensa de gastos. Realmente este malestar y furia mostrados por esta persona no son explicables, ya que si el motivo de la venta del DVD era, simplemente, poner al alcance de quien quisiera la revista al completo, sin ánimo de lucrarse, debería alegrarse de que alguien pusiera su empeño y trabajara hacia la misma dirección. Desde el momento en que Stalvs justifica punto por punto el precio de su oferta, viendo que no existe beneficio alguno, no entendemos qué molestia puede ocasionar el hecho de que otras personas pongan a disposición del público el mismo material.

Poco tiempo después Pedrete anunció que retiraba de la venta su DVD, ya que había escrito a Hobby Press para cerciorarse de que su iniciativa contaba con el correspondiente beneplácito de la editorial, pero Amalio Gómez le comunicó que, si bien se había autorizado la distribución de Microhobby en determinados sitios web (microhobby.org y WOS), bajo ningún concepto (y por política de empresa) se autorizaba la venta, ni a Pedrete como iniciativa particular, ni a ningún otro particular o sitio web, incluido microhobby.org (y, por ende, Stalvs) en formato DVD, CD o similar.

Como podeis ver, parece que nuestro mundillo se situa en el mismo triángulo de las Bermudas, sucesos extraños acontecen en cuanto no se lleva la razón y se sigue el juego en todo a determinadas personas.

Desde aquí unicamente nos queda felicitar efusivamente a Manuel por la ardua labor que ha llevado a cabo escaneando todas las revistas con una paciencia digna de elogio. A muchos nos ha hecho realidad un sueño y cualquier otra circunstancia en la cual se

quiera aprovechar para sacar beneficio, bien sea cobrado en medallas, económico o de cualquier otra índole, no deja de causarnos repulsión, malestar y tristeza.

TV Game o la esperanza de que se sigan haciendo juegos para Spectrum

Los húngaros de WSS, que no hace mucho nos deleitaron con la trilogía Flash Beer, sacaban al mercado hace breves fechas un nuevo juego: TV Game. Más que juego le podemos llamar experimento, ya que, basándose en juegos sencillos, como el PONG, ponen en práctica una serie de nuevas técnicas que permiten llevar al Spectrum un paso más lejos, enseñándonos efectos como la impresión en el borde de la pantalla, por poner un ejemplo.

Lejos de querer entrar en análisis sobre si el juego es bueno o malo, queremos hacer hincapié en el hecho de que se pueden seguir haciendo juegos y cobrar por ellos. Por el irrisorio precio de 4 euros en este caso, o poco más de 6 en el del Flash Beer, podemos disponer de títulos presentados como antaño, con su cinta, carátula e instrucciones. Y si queremos que se sigan haciendo, una buena forma de ayudar es invirtiendo esa pequeña cantidad de dinero. En el caso del TV Game, podemos descargarlo desde la WEB de WSS, pero en la cinta obtendremos como bonus una versión del juego para los Spectrum con 128K de memoria.

Para facilitarnos aun más la adquisición de estos títulos, Matranet los ha importado y los vende a precio de coste. Así no debemos preocuparnos por hacer pagos al extranjero, afrontar el alto coste del envío por correo y evitaremos las dudas que esto despierta en nosotros.

WOS sigue mejorando

Hace escasos días la página referencia de todo aficionado al Spectrum añadía un nuevo servicio: RANDOM.

Esta sección, de aparente simpleza, nos va a hacer pasar muy buenos ratos. Con un simple clic en un botón obtendremos al azar la ficha de un juego, de entre la enorme base de datos de que dispone. Los criterios de selección son muy sencillos y podemos adaptarlos a nuestro gusto. Para empezar, selecciona juegos que no estén M.I.A. o de distribución denegada por sus propietarios, y que tengan determinada puntuación y número de votos. Sencillo.

Desde este planteamiento tan simple, WOS nos dará a conocer juegos de los que no habíamos oído hablar y que, en muchos casos, nos sorprenderán. Así saldremos de la rutina en la cual terminamos todos, jugando a los mismos juegos de siempre.

Otra vertiente positiva de este apartado es la incitación que ejerce a obligarnos a participar votando a los juegos que probemos. Con este sencillo acto iremos dando opciones a los diferentes juegos para que salgan más o menos votados, y otras personas se guíen por esta puntuación para probarlos o desestimar su carga.

Un ejemplo de cómo desde el trabajo diario se puede llegar a la perfección. Increible WOS, increible Martijn.

El concurso sobre la boda

Con motivo de la boda real que nos "invadió" hace no mucho tiempo, Radastan convocó un concurso relámpago de juegos basados en la misma. Relámpago porque el plazo para presentarlos no superó las 48 horas. Aun así, se presentaron tres juegos a concurso con diferentes enfoques. Partiendo de que había poco tiempo se puede considerar un éxito la participación en el mismo.

La calidad de los programas no se puede decir que fuera elevada, pero desde la simpleza también se pueden hacer cosas interesantes. No vamos a entrar a comentar uno por uno los juegos, pero podéis entrar a la web de Radastan y juzgar por vosotros mismos.

Quizá el lado menos positivo del concurso fue el cambio de normas durante el desarrollo del mismo, algo a lo que, por desgracia, ya nos tiene acostumbrados Miguel Ángel desde el concurso de juegos en BASIC del año pasado. En este caso la prorroga en doce horas de la finalización del plazo para la presentación de programas que, si bien no influyó mucho, y favoreció la participación, si ofreció una sensación de poca seriedad y de acomodo "a lo que interese". El año pasado sucedió algo similar con el concurso de BASIC ya que, si bien, en un principio no se admitían juegos que tuviesen más gráficos que los creados en los UDG, luego se fue ampliando el margen con nuevas normas. Cosa que favoreció claramente a los más tardíos, en detrimento de las personas que presentaron los juegos al principio de concurso.

Si estos concursos son bienvenidos, aunque sea por el simple hecho de poder ver nuevo software, situaciones como éstas pueden hacer echar el freno a más de uno tentado de participar.

El hackeo de Speccy.org

Hace unas semanas, el portal speccy.org sufrió varios ataques cibernéticos. La "gracia" consistía en ejecutar un script contra Geeklog, el software que da soporte a todo el sistema de noticias, comentarios, encuestas, etc., inyectando comentarios en todas las noticias con enlaces a sitios pornográficos, *spam* y demás lindezas.

Todo esto debido a un *bug* en la versión de Geeklog instalada en speccy.org. El ataque no habría tenido mayores consecuencias si no fuera porque ocurrió justo durante el viaje de bodas del administrador de speccy.org, Santiago Romero. Al volver de viaje y encontrarse el panorama, su lógica reacción fue cerrar el "chiringuito", harto de niñerías que sólo buscan el daño por el daño. Afortunadamente, tras el calentón inicial, reconsideró su postura y procedió a actualizar el software para corregir dicho *bug*. Por otra parte, se han deshabilitado los comentarios de forma anónima en el portal.

Ignoramos si debido a esta restricción, desde entonces nadie ha aportado ningún comentario a las, ya de por si,

escasas noticias del portal. Tan solo ha habido comentarios en la encuesta. La verdad es que registrarse no cuesta nada, e identificarse tampoco es muy trabajoso. Es más, cabe la posibilidad de identificarnos una primera vez y quedar para siempre identificados desde la misma máquina/navegador desde donde nos conectemos a Internet. Quizás sea simplemente que estamos atravesando un período de vacas flacas.

En cualquier caso, y como conclusión de este triste episodio, queda constancia de qué fácil puede resultar destruir aquello que se construye con el trabajo y el esfuerzo de toda una comunidad. No se trata de "ceder al chantaje", como rezaban algunos comentarios que pudieron leerse con referencia al ataque. Hay que pensar que esto es una afición, proyectos en los que se colabora desinteresadamente y empleando tiempo libre. La paciencia y las ganas de esforzarse tienen límite. Como es lógico.

El peligro del vaporware

Últimamente han vuelto a saltar a la palestra varias iniciativas de software nuevo para Spectrum realizado en nuestro país. Las más significativas son Castlevania ZX, de Radastan, y ZX Fútbol ¿? (no sabemos si ése es su título definitivo), de Z80User.

Radastan ya debe estar acostumbrado a que periódicamente se le interrogue por el estado de su proyecto. Pero hasta cierto punto es algo normal. Él mismo se ha encargado de alimentar la expectación haciendo ver que el tema está controlado y va para adelante. De hecho, se ha permitido el lujo de rechazar la ayuda de varios colaboradores que se han ofrecido, bien a trabajar en los gráficos (el tema gráfico ya está cerrado), bien en el código (según Radastan, la inclusión de un nuevo programador no haría otra cosa que retrasar la tarea - desde aquí nos permitimos la licencia de dudar de su capacidad como organizador, y hasta de programador, basándonos en esas afirmaciones).

Por cierto, suponemos que el proyecto sufrirá un nuevo retraso, ya que hace breves fechas Radastan ha sido padre. Desde aquí, deseamos nuestra más sincera enhorabuena a los progenitores y una feliz existencia a la criatura.

Por otra parte, Z80User parece que está trabajando en un nuevo juego de fútbol. Según sus palabras, va a tener una parte de estrategia estilo PCFútbol, con un motor de simulación que se comportará, al menos, como Matchday. Desde luego que es un listón nada desdeñable. El sueño de cualquier jugón.

Puestos al habla con Z80User para conocer el estado real del proyecto, e intercambiar opiniones, no podemos por menos que comentar que quizás esté empezando la casa por el tejado. Está trabajando en cosas como la introducción de nombres reales, pantallas de la interfaz, diseño de los escudos de los equipos. Si bien todo ese trabajo requiere un gran esfuerzo que hay que valorar, puede que sea un esfuerzo baldío, ya que el proyecto carece de una planificación mínima sobre temas nada triviales, como pudiera ser qué concepto de simulación se quiere emplear (por turnos, jugada a jugada, etc.). Es bastante probable que, con un montón de trabajo hecho, luego las cosas no cuadren por esa falta de planificación inicial, lo cual llevaría comprensiblemente al desánimo de la persona o personas que estén trabajando en ello.

En nuestra opinión, ambos proyectos corren el serio peligro (por distintos motivos) de quedar en nada, lo cual sería una verdadera pena, y siempre supondría el recelo ante nuevas iniciativas que pudieran aparecer.

En verdad que somos los primeros que estamos deseando que aparezca nuevo software para nuestras entrañables máquinas, y si es de alguien cercano, mejor que mejor. Pero sí que es peligroso jugar con la ilusión de la gente, creando expectativas que puedan ser engañosas. Anunciar que se está trabajando en algo puede resultar un arma de doble filo, ya que puede llegar a crear la "obligación" de acabar ese proyecto. Tampoco se trata de llevar las cosas a un secretismo extremo, pero sí ser un poco cautos a la hora de hacer públicas según qué cosas.

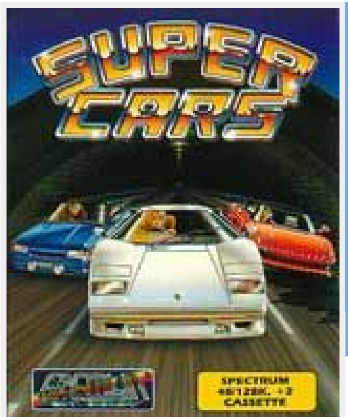
LINKS

- Microhobby.org: http://www.microhobby.org/
- El Trastero del Spectrum: http://www.speccy.org/trastero/
- SPA2: http://www.speccy.org/spa2/
- Sinclairmania: http://www.speccy.org/sinclairmania/
- La web de Pedrete: http://www.speccy.org/pedrete/
- WSS Team: http://www.c-system.hu/edy/weirdsciencesoftware/
- Matranet: http://www.matranet.net/
- WOS: http://www.worldofspectrum.org/
- Bytemaniacos: http://www.redeya.com/bytemaniacos/
- Z80User habla de su juego de fútbol: [groups.google.es] http://groups.google.es/groups?hl=es&lr=&ie=UTF-8&threadm=cal2q1$a5c$1@nsnmpen3-gest.nuria.telefonica-data.net&rnum=2&prev=/groups%3Fq%3Dfutbol%2Bgroup:es.comp.sistemas.sinclair%26hl%3Des%26lr%3D%26ie%3DUTF-8%26scoring%3Dd%26selm%3Dcal2q1%2524a5c%25241%2540nsnmpen3-gest.nuria.telefonica-data.net%26rnum%3D2



En este número, aparte del estupendo análisis que MIGUEL hace de la saga Wally en la sección Al Descubierto, FALVAREZ comenta un par de interesantes juegos: Super Cars y A Whole New Ball Game. Que los disfrutéis.

SUPER CARS



Título	Super Cars
Género	Arcade - Conducción
Año	1990
Máquina	48K-128K
Jugadores	1 Jugador
Compañía	Gremlin Graphics Software Ltd.
Autor	Spidersoft
Otros comentarios	

- Crash Issue 84
- Your Spectrum Issue 61

Desde siempre me han gustado los juegos de coches, no voy a negarlo. Y uno de mis títulos favoritos de la década de los 90 fue Super Cars II, en su versión para Commodore Amiga. Siempre me pregunté cómo sería la primera parte, y mira tú por dónde que WOS y su nuevo servicio Random vinieron a darme la respuesta en forma de juego para Spectrum: Super Cars.



Pantalla de carga

Super Cars es un juego de carreras de coches al más puro estilo arcade, con vista cenital (recordemos el clásico Supersprint). En este caso, el circuito excede el tamaño de la pantalla, por lo que un suave scroll nos acompañará a lo largo de las carreras.

Super Cars nos reta a llegar los primeros a lo largo de nueve circuitos de dificultad creciente. Curvas, rectas, cambios de rasante y cruces a distintos niveles son los elementos que conforman cada uno de ellos. Especialmente complicados resultan los túneles, ya que perdemos de vista a nuestro coche hasta que salgamos de ellos, con lo que si no entramos bien rectos, los choques contra los laterales de la calzada estarán a la orden del día. La sensación de profundidad está técnicamente bien resuelta gracias al uso de tramas y sombras.

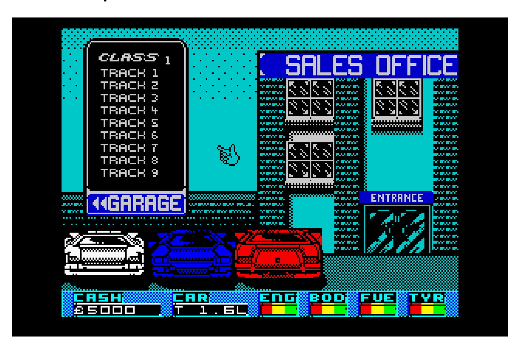


Menú versión 48K

Una vez hayamos completado los nueve circuitos, si hemos conseguido llegar entre los tres primeros en cada carrera, obtendremos un código y pasaremos al siguiente nivel. La dificultad aumentará, además de competir contra más coches controlados por el ordenador encontraremos obstáculos en la pista, como puede ser barro, aceite o agua.

El objetivo del juego consiste en ganar la mayor cantidad de dinero posible en la competición, para así adquirir mejoras para nuestro coche (o un coche nuevo), o bien para reparar los desperfectos que hayamos ocasionado durante la carrera. Tendremos que vigilar el estado de nuestro motor, carrocería, combustible y ruedas, si no queremos quedarnos tirados en medio de la dura pugna por

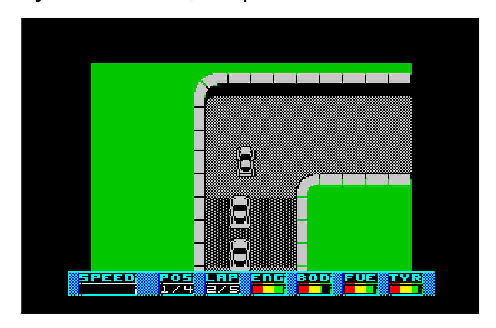
las primeras posiciones.



Menú versión 128K

El juego se editó en versiones para 48K y 128K. La primera diferencia palpable entre ambas versiones, aparte de la consabida multicarga para la de 48K (la versión para los hermanos mayores carga todo del tirón), radica en el aspecto de los menús. En la versión de 128K, controlamos un puntero para seleccionar las distintas opciones, dando un aspecto más "actual" a la interfaz. No obstante, la funcionalidad es idéntica en ambas versiones.

El manejo se puede hacer mediante joystick Sinclair, Kempston o teclado. La opción de redefinir teclas no está contemplada, aunque están permitidas muchas combinaciones, entre ellas las habituales (OPQA). Durante la carrera aceleraremos con el botón de disparo, giraremos con las direcciones laterales (izquierda y derecha), y arriba y abajo los usaremos para lanzar misiles hacia delante y hacia atrás, respectivamente.



Vamos los primeros, pero nos siguen de cerca

En cuanto al sonido, como de costumbre encontramos diferencias entre las versiones de 48K y 128K. La primera de ellas carece de melodías tanto en los menús como durante el juego, y el sonido se limita a los efectos durante la carrera, como son derrapes, choques o disparos. Por otra parte, podremos disfrutar de melodías en la versión 128K, tanto en los menús como en el transcurso de las carreras. Eso sí, el reparto de los 3 canales de sonido no ha sido muy afortunado, ya que al reproducir algún efecto la música se entrecorta, produciendose una sensación bastante extraña.

En resumen, un juego bien hecho, partiendo de una idea sencilla, muy cuidado técnicamente y no demasiado difícil, que nos enganchará hasta que completemos todas las carreras y nos hagamos con todos los vehículos. Se echa en falta un modo de dos jugadores para poder competir con un amigo, pero quizás eso excediera la capacidad técnica de nuestras máquinas para mover un scroll tan suave (en esta ocasión).

Valoraciones

Originalidad:	[6]	
Gráficos:	[8]	
Sonido:	[6]	
Jugabilidad:	[8]	
Adicción:	[8]	
Dificultad:	[7]	

Trucos:

Puedes encontrarlos en The Tip Shop

Descárgalo de:

- WOS

A WHOLE NEW BALL GAME



Título	A Whole New Ball Game
Género	Puzzle
Año	1989
Máquina	48K
Jugadores	1 Jugador
Compañía	Crash
Autor	Pete Cooke

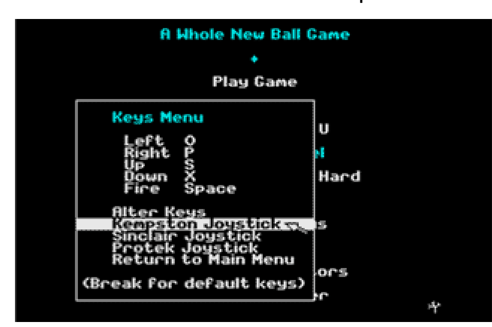
Hacia las postrimerías de la vida comercial del Spectrum, las revistas comenzaron a adjuntar cintas en las que se incluían, como norma general, demostraciones de próximos lanzamientos, títulos descatalogados y, como en este caso, juegos originales. En esta ocasión hemos pulsado nuevamente la opción *Random* de WOS y hemos encontrado este puzzle que, al menos para quien escribe este artículo, resulta totalmente novedoso (que no original).

El objetivo del juego es sencillo. Se trata de capturar unas píldoras amarillas de energía que encontraremos en la pantalla. Para ello, haremos pasar una bola por encima. La gracia está en que nosotros no controlamos directamente el movimiento de la bola, sino que debemos colocar en la pantalla unos deflectores para modificar su camino y hacer que siga el que nosotros queramos. Para completar cada pantalla, debemos capturar todas las píldoras antes de que se acabe el tiempo, o perderemos una vida. Asimismo, debemos estar atentos a otro tipo de elementos que puedan alterar el curso y la velocidad de la bola (el juego los llama "Effectors", no he sabido cómo traducir este término).



La pantalla de carga

El control del título, tanto en los menús como en el juego propiamente dicho, se efectúa a través de un puntero, como si se tratara de un ratón. Si dispusiéramos de tal periférico, pues sería muy cómodo, pero no es el caso, así que lo moveremos con el teclado (redefinible), o bien un joystick de protocolo Kempston, Sinclair o Protek. En cualquier caso, al menos bajo mi punto de vista, al final se hace bastante incómodo mover el puntero.

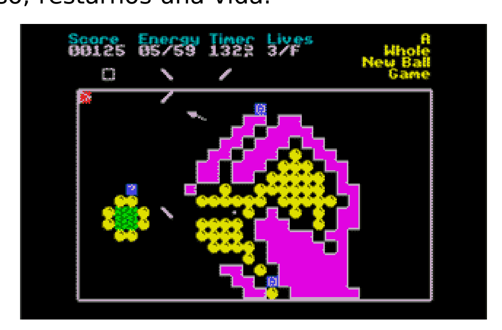


El menú de selección de controles

En el menú podremos modificar otros parámetros, como son la pantalla en la que empezaremos (a elegir entre cinco), el nivel de dificultad (fácil, moderado o difícil), ver las puntuaciones, modificar el método de control (entre los anteriormente descritos), consultar las reglas, la descripción de los "Effectors" y, por último, un diseñador de pantallas.

Abajo a la derecha, como curiosidad, aparece el símbolo de un árbol. Si pulsamos en él, accederemos a un generador de árboles fractales. Interesante... Eso sí, cuidado con los parámetros que eliges, porque luego no se puede interrumpir la generación del fractal.

Los "Effectors", como hemos comentado anteriormente, se encargan de modificar la trayectoria y la velocidad de la bola. No vamos a entrar a describir exhaustivamente cada uno de ellos (se puede consultar una descripción detallada en el menú de opciones), simplemente comentar que pueden girar la bola 90 grados, obligar que siga una dirección determinada, dirigirla hacia un lugar aleatorio, teletransportarla a otro lugar de la pantalla, acelerarla, frenarla, bloquear su camino e, incluso, restarnos una vida.



Devorando cápsulas de energía...

El juego se hace complicado ya que, aparte del hecho de tener que colocar los muros deflectores para que la bola vaya por donde queremos, tendremos que apañarnos con un método de control que es, a todas luces, inadecuado para su manejo con el teclado o un joystick. Enlentece mucho nuestros movimientos y los hace poco precisos, en momentos donde la rapidez y la precisión son cruciales.

En cuanto al sonido, la pantalla de presentación tras la carga viene amenizada por una conocida melodía de Mozart, en el formato monofónico clásico del Spectrum. Es de agradecer que la cosa quede ahí, ya que resultaría excesivamente pesada. El resto del juego se limita a efectos sonoros en los rebotes de la bola.

En resumen, una vuelta de tuerca más a una idea ya trabajada (estoy recordando ahora mismo Deflektor, del genial Costa Panayi), con una realización técnica simplemente correcta, que nos hará pasar un buen rato si no acabamos desquiciados con la forma de controlar el juego.

Valoraciones

Originalidad:	[7]	
Gráficos:	[6]	
Sonido:	[6]	
Jugabilidad:	[6]	
Adicción:	[7]	
Dificultad:	[9]	

Trucos:

Puedes encontrarlos en The Tip Shop

Descárgalo de:

- WOS

FALVAREZ



FALVAREZ

ADAPTANDO PERIFÉRICOS AMSTRAD AL SPECTRUM.

Volvemos a casa exultantes, con nuestro +2 recién adquirido debajo del brazo. "Por fin un ordenador de verdad". La membrana de nuestro añorado gomas se cae a cachos, demasiadas partidas al Videolimpic. Guardamos el Spectrum de 48K en un cajón (nuestra santa madre tiró la caja hace tiempo, tantos trastos estorban en casa) del que probablemente nunca vuelva a salir hasta la próxima mudanza (que tiemble, los hay que acabaron en un contenedor dentro de un petate de la mili con el resto de la colección). Enchufamos los cables nuevos, recogemos también el Computone (tras encomendarnos a San Azimut) y, mientras cargamos nuestros flamantes juegos para 128K (una música en el menú, vaya cambio), conectamos nuestro Quick Shot II en el puerto lateral. Llega la hora de comenzar la partida y... ¿qué pasa? ¡El joystick no responde! ¿Estará roto? O, pensándolo mejor, ¿no será que Alan Sugar nos la ha vuelto a jugar?

Pues sí. En su afán de colocarnos periféricos nuevos (y aflojarnos el bolsillo), Amstrad decidió cambiar la disposición de pines en los puertos de joystick, de tal forma que sólo se podían usar los mandos SJS. Así que, en nuestro artículo de este número, vamos a realizar unos sencillos adaptadores que nos permitirán, por un lado, usar joysticks de norma Atari en nuestros +2/+3 y viceversa, esto es, si tenemos algún mando SJS, poder usarlo con nuestros interfaces Kempston, Sinclair u otros. Además, como pequeña venganza, vamos a realizar un adaptador a nuestro cable RGB-Euroconector para poder conectar el +2/+3 a un monitor Amstrad CTM644 en color, para que el pobre se entere de una vez de lo que se estaba perdiendo siendo usado exclusivamente por los modelos CPC.

No se trata de montajes difíciles, ninguno de los tres. Sólo se trata de recablear pines entre los conectores, para que las señales lleguen a los sitios adecuados.

MATERIAL NECESARIO

Para cada uno de los adaptadores de joystick:

- Clavija DB9 macho para montaje aéreo (y su carcasa correspondiente).
- Clavija DB9 hembra para montaje aéreo (y su carcasa correspondiente).
- Cable de 6 hilos, tan largo como queramos hacer el "adaptador".
- Soldador y estaño.
- Pelacables (opcional).
- Destornillador (para cerrar la carcasa).

Para el adaptador RGB al monitor Amstrad:

- Clavija SCART hembra para montaje aéreo (Imagen 1).



Imagen 1. Clavija SCART hembra aérea

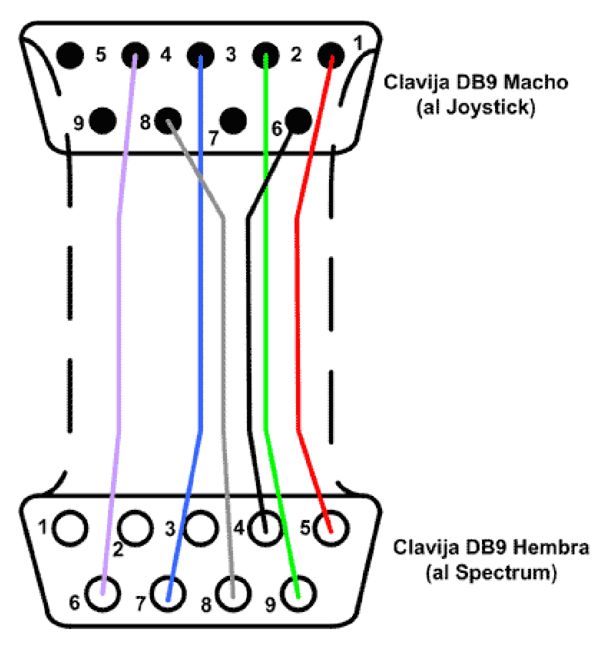
- Clavija DIN 6 hembra para montaje aéreo (y su carcasa correspondiente).
- Clavija mini-jack estéreo hembra para montaje aéreo.
- Cable de 6 hilos, tan largo como queramos hacer el "adaptador".
- Cable doble de hilo+malla (el típico que llevan los cascos, por ejemplo), de la misma longitud aproximadamente que el anterior.
- Soldador y estaño.
- Pelacables (opcional).

PROCEDIMIENTO

El procedimiento es tan sencillo como seguir el esquema correspondiente para cada uno de los tres adaptadores:

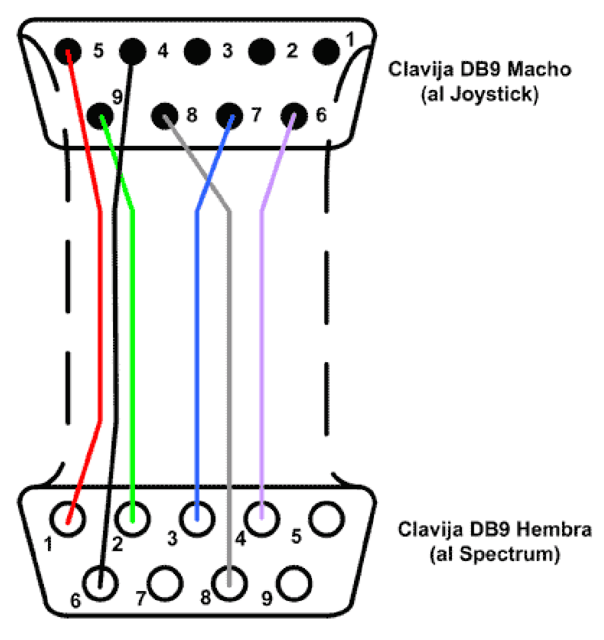
- Adaptador Atari a SJS - Esquema 1.
- Adaptador SJS a Atari - Esquema 2.
- Adaptador RGB-SCART a Monitor Amstrad CTM644 - Esquema 3.

Como siempre, en los esquemas representamos las clavijas desde el lado de las soldaduras.



Esquema 1. Atari a SJS

En el caso de los joysticks, tenemos que recablear las señales Arriba, Abajo, Izquierda, Derecha, Disparo y Masa. Un adaptador es complementario del otro. Si conectamos los dos seguidos, las señales no cambiarán de pin.

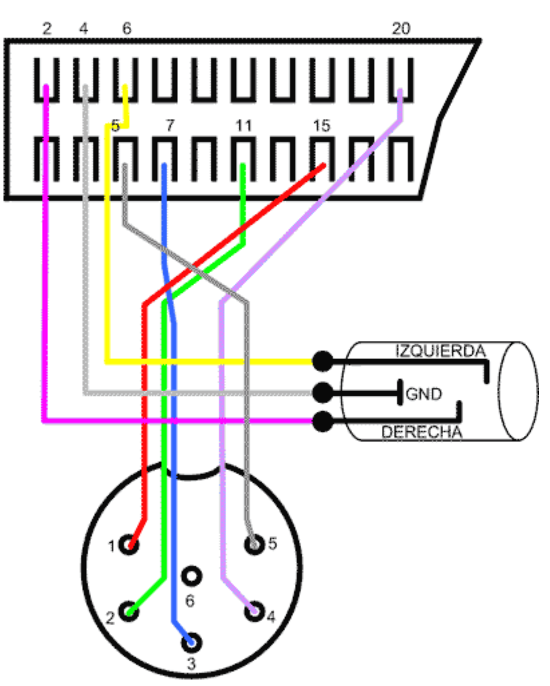


Esquema 2. SJS a Atari



Imagen 2. Detalle de las soldaduras en la clavija DB9

En el caso del monitor, tenemos que recablear las señales Rojo, Verde, Azul, Sincronismo y Masa. Recordemos que el monitor no tiene altavoces, así que la señal de audio la llevaremos a la clavija hembra mini-jack, para conectar unos altavoces autoamplificados, por ejemplo.



Esquema 3. RGB-SCART a Monitor Amstrad CTM644

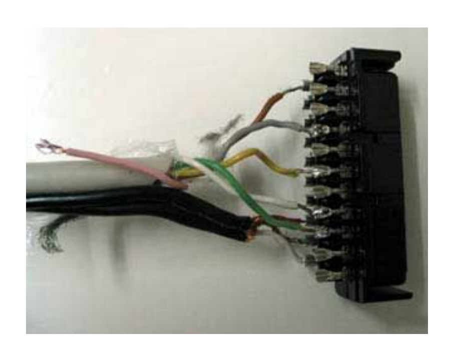


Imagen 3. Detalle de las soldaduras en la clavija SCART

En la Imagen 4 vemos uno de los dos adaptadores de joystick que acabamos de construir. La longitud del adaptador, a gusto del consumidor. Nostros los hemos construido algo larguitos, para no tener problemas de tiranteces cuando estemos jugando. Por cierto, que si los vamos a usar en un +2, no debemos poner la carcasa en la clavija hembra, ya que tropieza con el ordenador y no entra en su sitio (al menos con el tipo de carcasas que hemos usado. El +3 no tiene ese problema.

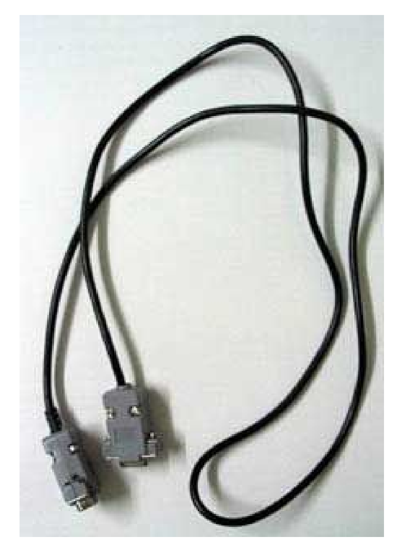


Imagen 4. Adaptador para joysticks

En la Imagen 5 vemos el adaptador RGB. La clavija mini-jack, que va aparte, la podemos enchufar a unos altavoces de sobremesa de PC, por poner un ejemplo.



Imagen 5. Adaptador RGB para monitor CTM644

El adaptador RGB sirve de puente entre el cable RGB que construimos en el número 3 de Magazine ZX y el monitor Amstrad.



Imagen 6. El adaptador conectado al cable RGB del +3

Para finalizar, unas partiditas al Match Day en nuestro monitor Amstrad nunca vienen mal. También vemos el cable RGB y la disquetera de 3"1/2 dando servicio.



Imagen 7. Nuestro +3 conectado al monitor Amstrad

LINKS

- Pinout Joystick Atari http://www.gamesx.com/hwb/co_JoystickAtari2600.html
- Pinouts de conectores Amstrad http://andercheran.aiind.upv.es/~amstrad/docs/connect.html
- Pinouts de conectores Sinclair y sus clones http://ci5.speccy.cz/cygnus/connect.htm
- Hilo acerca de adaptar joysticks [groups.google] http://groups.google.es/groups?hl=es&lr=&ie=UTF-8&threadm=slrnb1o3g7.khv.sromero@pinsa.escomposlinux.org&rnum=9&prev=/groups%3Fq%3Djoystick%2Bgroup:es.comp.sistemas.sinclair%2Bauthor:Santiago%2Bauthor:Romero%26hl%3Des%26lr%3D%26ie%3DUTF-8%26scoring%3Dd%26selm%3Dslrnb1o3g7.khv.sromero%2540pinsa.escomposlinux.org%26rnum%3D9



NÉSTOR LUCAS

LA SAGA WALLY, PATAS ARRIBA

Desde el principio de la historia de los videojuegos, éstos han tenido protagonistas con más o menos carisma. Algunos de ellos han llegado a protagonizar auténticos "culebrones", lúdicamente hablando por supuesto. En España tuvimos a nuestro Johnny Jones, creado por Dinamic, con sus Saimazoom, Babaliba y Abu Simbel Profanation, y se nos quedó a medio camino del helado polo. Codemasters nos obsequió con varios de estos héroes como eran el huevo Dizzy y su decena de juegos, uno más uno menos, al elefante DJ o a Seymour y sus aventuras de cine. Ultimate y su Sabreman, y un largo etcétera hasta llegar al personaje que nos ocupa hoy, Wally Week, al que conocimos de la mano de Mikro-Gen.

¿Que quién es Wally Week? Wally es el típico inglés de clase obrera, por el día trabaja en una planta de montaje de coches muy peculiar, o eso nos hacen saber en su primer juego, y por la noche es habitual de los pubs y tiene fama de ser un bebedor de cerveza infatigable (eso nos lo hace saber su oronda tripa).

Meet the Gang!
Wilma
Tom
Dick
Harry
Herbert
"Everyone's a Wally"
1ST EVER MULTI-ROLE ARCADE ADVENTURE
MIKRO-GEN

Wally's family and friends

Wally tiene familia: su esposa Wilma y su hijo Herbert, el demonio hecho niño, infante que protagonizará él solito una de las partes de la serie; y amigos muy peculiares, Dick, Tom y Harry, cada uno con su profesión y sus "pintas". A todos estos personajes los iremos conociendo a lo largo de los distintos juegos que componen la serie, cinco en total.

Aunque la serie "Wally" es conocida por estar enmarcada en el género de las videoaventuras, lo cierto es que el primer juego con el que se nos da a conocer al personaje, Automania, es un plataformas puro y duro. Este Automania lo programó Chris Hinsley en el año 1984, quien fue el mismo autor de su continuación, Pyjamarama, también programada en el mismo año, y de la secuela de éste y el mejor juego de la saga para el criterio de quien esto escribe: Everyone's A Wally, creado en el año siguiente, 1985. Estos dos últimos títulos ya son dos videoaventuras en toda regla. Para los dos juegos finales Mikro-Gen cambió de programador, aunque realmente no se aprecian grandes diferencias y los juegos siguen siendo de lo mejor en su campo. David J. Perry fue la persona que en el año 1985 creó el Herbert's Dummy Run, en el cual dejábamos por el momento de controlar a nuestro barrigudo amigo Wally y tomábamos el control de su hijo Herbert. Y, para finalizar esta espectacular serie, llegó en el año 1986 el Three Weeks In Paradise, y nuestro amigo Wally convertido en héroe a la fuerza al rescate de su familia. Este juego fue realizado al alimón por David J. Perry y Nick Jones.

En el período de dos años largos que tardaron en salir los cinco juegos se fueron introduciendo mejoras y características que luego serían habituales en muchos títulos. Entre otras, Everyone's A Wally fue el primer juego en el cual podíamos manejar a varios personajes alternativamente y cuyo concurso era necesario para poder terminar la aventura.



Chris Hinsley, el padre de Wally. No se aprecia el parecido

En Pyjamarama se incluyó un subjuego tipo Invaders, al igual que otro tipo Asteroids en el Everyone's, por poner algunos ejemplos.

Los cuatro últimos juegos hicieron gala de un surrealismo muy acentuado que, si bien se refleja con más claridad en el Pyjamarama y su pesadilla, lo tenemos presente en toda la serie con el uso ilógico de objetos. Esto hacía que, para terminar un juego, tuvieramos que buscar el uso más absurdo para cada ítem. Realmente sin las guías que nos ofrecían las diferentes publicaciones de la época muy poca gente habría terminado cualquiera de ellos.

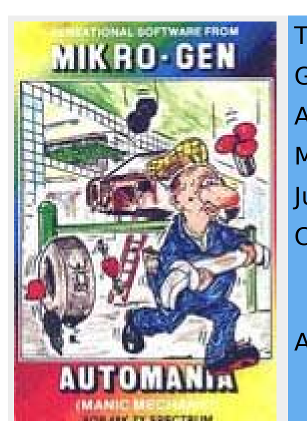


David Perry, programador de las dos últimas entregas de la saga

Seguid leyendo, en las próximas líneas vamos a dar repaso, una por una, a estas cinco maravillas de la programación lúdica.

[Nota] Todas las imágenes han sido sacadas de WOS y la web personal de David Perry.

Automania

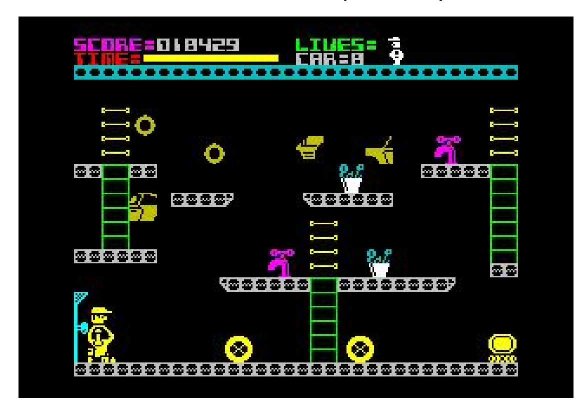


Título	Automania
Género	Plataformas
Año	1984
Máquina	48K
Jugadores	1 Jugador
Compañía	Mikro-Gen
Autor	Chris Hinsley

En su primera aparición en las pantallas nuestro amigo Wally eligió un juego plataformas para que fuéramos conociéndole. En este juego tenemos que ayudar a Wally en su trabajo: el montaje de coches en una fábrica. Esta tarea parece rutinaria, si no fuera por la insistencia que tienen las ruedas, bielas y otras partes integrantes de los coches en rebelarse contra nosotros e impedirnos realizar nuestra labor correctamente.

El juego se divide en dos pantallas. La primera de ellas a la derecha, siendo ésta la pantalla de inicio, en la que tenemos los coches divididos en seis piezas y que irá cambiando y subiendo su dificultad según vayamos completando fases. La otra, a la izquierda de la anterior, en la que deberemos depositar las partes, y que siempre será igual salvo los enemigos que nos ataquen.

La mecánica es sencilla: en la pantalla de almacenaje recogemos una de las piezas del coche. Sólo podemos llevarlas de una en una, y en la pantalla de montaje las colocamos en el lugar del coche que correspondan. Pero claro, esto que parece sencillo nos lo tenían que complicar.



¿Te parecía difícil montar el primer coche?

Para llegar a cada una de las piezas tenemos que esquivar dos ruedas y un artefacto que se mueven por toda la zona baja y que tratan de golpearnos, ir saltando diferentes obstáculos como latas de aceite

y herramientas varias que están dispersas por el suelo del almacén y que tendremos que saltar o nos harán perder el equilibro y que caigamos desde lo alto al suelo. Por si esto fuera poco, las mismas plataformas por las que debemos desplazarnos se mueven en muchos casos y tendremos que calcular muy bien nuestros movimientos si no queremos encontrarnos con que, al finalizar el salto, ya no tenemos más que vacío en lugar de suelo y caeremos irremediablemente hasta la parte inferior. Pero aún hay más, debemos llevar las piezas en un tiempo límite, que será restablecido al máximo cada vez que coloquemos la pieza en su destino. Una vez que nos hemos hecho con una de las piezas debemos dirigirnos a la pantalla de montaje y colocar la pieza en el lugar correcto del coche, incluso saltando sobre él si es necesario. En esta pantalla, además de las ruedas y la máquina que intentan atropellarnos, nos caerán llovidas de las dos cintas de transporte diferentes piezas de coche que deberemos esquivar. Cuando completemos un coche con sus seis piezas pasaremos de fase, incrementándose la diicultad del juego en la pantalla de almacenaje (como comentábamos antes), pero como contrapartida nos recompensarán con una vida extra, que doy fe de que hará mucha falta, ya que la colisión con cualquier enemigo, caída o finalización del tiempo disponible decrementarán en uno el contador de las vidas. También nos cambiarán el modelo de coche que debemos montar, variando entre un Mini, Porsche, un taxi londinense, un Dyane 6 y muchos más a cual más atractivo.



No, no te van a dar el Porsche

Los marcadores de la pantalla, situados en la parte superior de la misma, son claros y concisos, indicándonos el tiempo restante que nos queda para llevar la pieza a su sitio correcto, el número de vidas que tenemos disponibles, la puntuación y los coches terminados que hemos conseguido por el momento.

Wally responde muy bien a los controles, que podemos redefinir a nuestro gusto, así como seleccionar joysticks con norma Sinclair o Kempston. Es de agradecer que el control sea preciso pues en este juego es imprescindible calcular el salto al milímetro o daremos con nuestros huesos en el duro suelo y veremos como colocan la lápida sobre nosotros.

Los gráficos son muy buenos, tanto en calidad como en colorido y tamaño. Nuestro protagonista, además de tener un tamaño considerable, está bien realizado, sentando las bases para sus secuelas con ligeras modificaciones. Los enemigos se ven claramente y no tendremos dudas de sus intenciones. La única pega que le encuentro en este apartado es el amasijo de pixels que se forma cuando transportamos una pieza del coche, y si encima subimos o bajamos por una escalera mejor no intentar adivinar qué es "eso" que se mueve por la pantalla.



Pantalla de carga

Una precaución que debes tomar si no quieres terminar odiando la música es anularla en el menú antes de comenzar la partida, durante el juego no podrás hacerlo. El tema está bien realizado pero para escucharlo una o dos veces a lo sumo, es el típico de las películas en blanco y negro de Laurel y Hardy. Muy machacón. Respecto a los efectos de sonido, los justos y limitándose a algún ruido al coger o dejar una pieza o al morir.

Resumiendo, podemos decir que Wally entró con muy buen pie en el mundo de los videojuegos, aunque luego cambiase las plataformas por las videoaventuras, muy acertadamente diría yo. Este Automania es un juego muy entretenido y adictivo que va a hacer que pasemos mucho tiempo intentando ver qué modelo de coche aparecerá en la siguiente fase e intentando enlazar esos tres saltos consecutivos correctamente para alcanzar aquella pieza que parece inaccesible a nuestro curtido trabajador.

Valoraciones

Originalidad:	[7]	
Gráficos:	[8]	
Sonido:	[6]	
Jugabilidad:	[9]	
Adicción:	[10]	
Dificultad:	[8]	

Trucos:

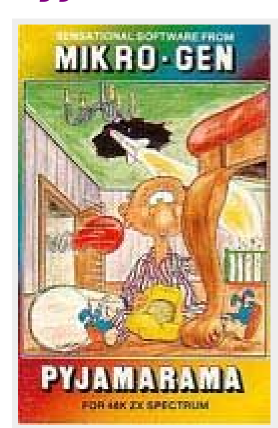
Puedes encontrarlos en The Tip Shop
http://www.the-tipshop.co.uk/cgi-bin/search.pl?name=Automania

Descárgalo de:

- WOS

http://www.worldofspectrum.org/infoseek.cgi?regexp=^Automania$&pub=^Mikro-Gen+Ltd$

Pyjamarama



Título	Pyjamarama
Género	Videoaventura
Año	1984
Máquina	48K
Jugadores	1 Jugador
Compañía	Mikro-Gen
Autor	Chris Hinsley

¿Cómo es tu peor pesadilla? ¿Te rodean cientos de arañas peludas? ¿Caes y caes y caes y sigues cayendo en una fosa sin fondo? ¿Te invitan a pasar toda una noche con Leticia Sabater? ¿Te persiguen y, por más que corres, no puedes dejarlos atrás? ¿Esperas jugar al Castlevania ZX antes de la jubilación? Todo esto son tonterías comparado con lo que va a pasar nuestro pobre amigo Wally en ésta, su primera videoaventura, y segunda aparición en el mundo de los juegos de ordenador.



Cargando el juego

Pyjamarama es el comienzo de una gran serie de juegos, el avance de un género que nos llegaría a deleitar con grandes títulos: la Videoaventura.

Wally se acostó como cada noche, después del recorrido habitual por los pubs del barrio, pero en esta ocasión se le olvidó poner el despertador para levantarse pronto e ir a trabajar por la mañana. Para colmo de males, una pesadilla acecha sus sueños, y no podrá salir de ella hasta que consiga poner en funcionamiento el despertador, cosa que tendrá que hacer en su mal sueño. Para seguir con su mala suerte, en su pesadilla Wally mide la mitad de su altura habitual, con la consiguiente dificultad para alcanzar objetos y sitios a los que llegaría con toda facilidad en su estado habitual. Este argumento, surrealista a más no poder, va a dar mucho juego.

Nuestro álter ego no podía vivir en una casa normal, como todos nosotros, con su bañito y sus dos o tres habitaciones, no. Tenía que vivir en una mansión de varias plantas, sótano, azotea y, para desgracia nuestra, repleta de seres extraños por obra y gracia de la pesadilla.

Con el objetivo claro, despertar a Wally, comenzamos la aventura. Pyjamarama está formado por un número elevado de pantallas, más de treinta, que deberemos recorrer recogiendo objetos que usaremos en otras localizaciones para conseguir objetivos determinados, y esquivando numerosos enemigos, que harán descender de manera alarmante nuestro nivel de energía, representado por un vaso de leche que va vaciando su contenido al menor roce con los indeseables inquilinos de nuestra morada.



¿Seremos capaces de despertarle?

Imaginar el uso de cada objeto es una tarea bastante difícil, ya que pensar en la utilidad que se le puede dar a un casco de fútbol americano dentro de una casa, por decir un ejemplo claro, es bastante complicado. Esto quiere decir que terminar el juego sin ayudas es una tarea para la que deberemos exprimir al máximo nuestra imaginación. Poca gente lo habrá terminado (yo entre ellos) sin utilizar las guías que venían en las revistas de la época. Los objetos son muy variados, desde un cubo, pasando por unas tijeras, una tarjeta perforada, etc., y muchas veces deberemos recorrer la casa de punta a punta para utilizar cualquiera de ellos donde lo requiere la situación, ya que no podemos llevarlos todos con nosotros, tendremos con conformarnos con portarlos de dos en dos.

El apartado gráfico del juego es sobresaliente. Todas las pantallas desbordan colorido y están cargadas con un alto número de detalles, además

de estar bien diseñadas para que no lleguen a cargarse excesivamente. Los gráficos tienen un tamaño bastante grande y se mueven con fluidez, tanto los enemigos como el personaje principal. Wally es una evolución del personaje ya conocido en Automania, esta vez con pijama y gorrito de dormir.

En cuanto al sonido, correcto para el tipo de programa que es. En el menú de opciones tenemos un conocido y machacón tema, pop corn, bien realizado. Durante el juego efectos variados al recoger objetos, golpearnos con los enemigos y poco más. Lo suficiente. Como curiosidad cabe decir que salieron dos versiones del juego. En la primera la música del menú era la que indicábamos unas líneas más arriba, pero en la segunda esa música era sustituida por otra de realización propia. ¿Problemas con los derechos de autor? Posiblemente.



Huevo de pascua: Estilo Invaders

Jugabilidad, toda, y un poco más. El juego es muy difícil, pero no por el manejo del personaje, que responde perfectamente a nuestros requerimientos mediante el teclado o el joystick. El movimiento es muy sencillo: desplazarse para ambos lados y saltar. Para recoger los objetos basta con pasar por encima de ellos cambiando uno por otro si llevamos los dos huecos ocupados. En cambio, es complicado a rabiar por el uso que tiene cada objeto. Premio para quien los adivine sin usar guías. Como huevo de pascua nos incluyen una especie de "Invaders" al llegar a una habitación determinada, con el cual podremos relajarnos y soltar nuestra fustración viendo desaparecer enemigos al dispararles con armas tan peregrinas como tenedores, botellas, etc. Estas sorpresas se repetirían en los futuros juegos de la saga e, incluso en algunos casos, serían necesarias para poder avanzar.

Pyjamarama es un juego muy adictivo, bien realizado, al que le falta poco o nada para ser uno de los mejores y una auténtica demostración creativa de su autor. Cuando lo jugué en su día pensaba que no se podría mejorar facilmente, pero la saga estaba en sus comienzos. Quedaba mucho por ver... y si no te lo crees, sigue leyendo.

Valoraciones

Originalidad:	[10]
Gráficos:	[9]
Sonido:	[7]
Jugabilidad:	[9]
Adicción:	[10]
Dificultad:	[10]

Trucos:

Puedes encontrarlos en The Tip Shop

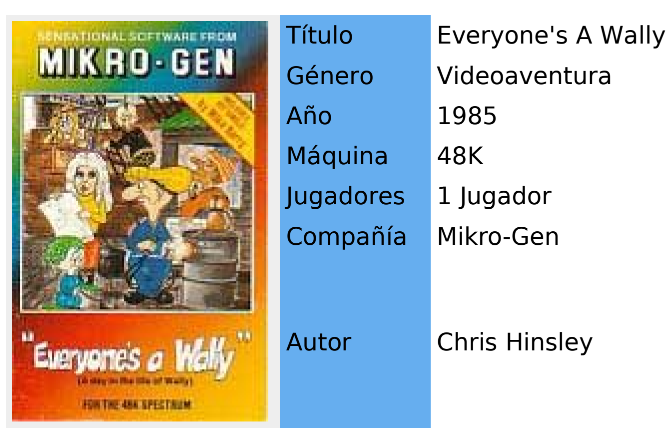
http://www.the-tipshop.co.uk/cgi-bin/search.pl?name=Pyjamarama

Descárgalo de:

- WOS

http://www.worldofspectrum.org/infoseek.cgi?regexp=^Pyjamarama$&pub=^Mikro-Gen+Ltd$

Everyone's a Wally



Título	Everyone's A Wally
Género	Videoaventura
Año	1985
Máquina	48K
Jugadores	1 Jugador
Compañía	Mikro-Gen
Autor	Chris Hinsley

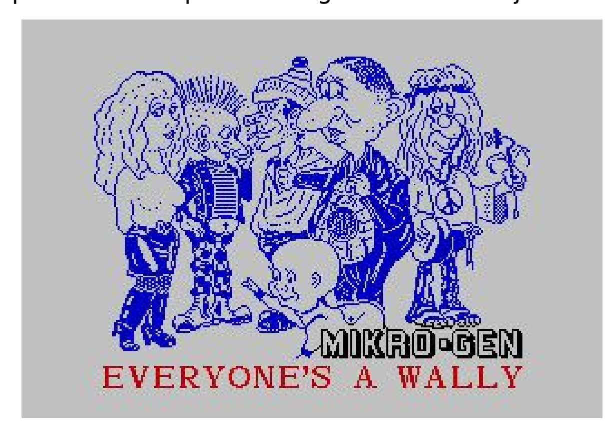
Cuando salió el juego que nos ocupa ahora, Everyone's A Wally: A day in the life of Wally, me frotaba los ojos ante lo que estaba viendo. Un juego que te permitía manejar cinco personajes, alternativamente, con total libertad de movimientos por las diferentes y coloridas pantallas que representaban un pueblo, y cuyo concurso era imprescindible para terminar la aventura.

Visto ahora, pasados veinte años desde su creación, nos puede parecer algo pueril, pero Everyone's marcó el punto álgido en las

videoaventuras, un juego imitado hasta la saciedad pero muy pocas veces, por no decir ninguna, superado.

Y después de esta introducción, en la que queda claro que al que esto escribe le encanta el juego, voy a intentar ser lo más objetivo posible a la hora de analizarlo.

En esta entrega de la saga Wally nos dan a conocer a su familia y al grupo de amigos, variopinto, con los que se junta nuestro protagonista. La mujer de Wally se llama Wilma y juntos tienen un revoltoso hijo, Herbert, que no hará más que incordiarnos durante toda nuestra aventura. Los tres amigos de Wally son Tom, Dick y Harry. Todos ellos van a ser los protagonistas de esta aventura. Aventura que comienza con un objetivo claro: abrir la caja fuerte del banco para conseguir dinero y repartirlo entre el grupo. Para ello tendremos que llevar a buen término multitud de reparaciones por toda la ciudad con el fin de conseguir unos libros diseminados por la misma y llevarlos a la biblioteca, con lo que obtendremos la combinación para abrir la caja fuerte. Cada personaje tiene una profesión específica de la que deberemos aprovecharnos para conseguir nuestros objetivos.



La pantalla de carga

El concurso de cada personaje es vital, a excepción del crío Herbert, que se va a dedicar toda la aventura a gatear recorriendo la ciudad, incordiándonos en nuestra labor ya que, al más mínimo roce, la energia del personaje que estemos manejando en ese momento disminuirá drásticamente. Cada amiguete de Wally, y el mismo Wally cuando no le manejemos, estará recorriendo el pueblo sin parar, y cuando tengamos que cambiar el manejo de uno a otro necesitaremos localizarle con el personaje actual. Esto se puede complicar bastante, pero tenemos una ayuda útil: cada personaje del juego está identificado con una tecla del 1 al 5, de manera que si pulsamos la tecla y el personaje está en pantalla, cambiaremos el manejo a él, liberando al que manejáramos hasta este momento. Si no está en el mismo escenario, un rótulo nos indicará su situación, con lo cual necesitaremos ir siguiéndole para poder intercambiar los controles. Este seguimiento no es tarea sencilla, ya que el colega no piensa estarse parado, y necesitaremos mirar con frecuencia su situación para localizarle. Un conocimiento exhaustivo del mapeado nos será imprescindible en estas situaciones. Para incrementar un poco (bastante) la dificultad, a nuestros amigos les gusta coger los objetos en sus largos paseos y cambiarlos de lugar, imaginad el resto.

Con lo dicho anteriormente podéis imaginar que el juego es muy complicado, y no os equivocáis. Adivinar el uso que tenemos que darle a cada objeto, usarlo, averiguar qué personaje lo debe usar, esquivar a Herbert y a los diferentes obstáculos que minarán nuestra salud es una tarea ardua pero satisfactoria cuando consigues ir resolviendo los diferentes puzzles a los que la pandilla se enfrentará.



El punto de partida con todos los protagonistas

El juego, gráficamente hablando, es una pequeña obra de arte. La ciudad está muy bien representada, encontrando edificios y locales habituales como pueden ser un pub, la estación de ferrocarril, el puerto, las alcantarillas y un largo etcétera de localizaciones diferentes, tanto exteriores como interiores. Todas las pantallas están realizadas con un alto grado de detalle y colorido. Los personajes están muy bien diseñados, algo a lo que nos tenían acostumbrados con Wally, que en esta ocasión evoluciona un poquito más. Wilma, delicada ama de casa; Tom, el punky mecánico; Harry, un hippie "chispas" y Dick el rudo fontanero. Todos ellos se mueven con suavidad y el manejo es sencillo, limitándose al movimiento lateral y salto, cosa que se consigue con tres teclas, más una cuarta para entrar en los diferentes edificios y tomar calles perpendiculares a la que estemos. Como en las entregas anteriores, y algo que seguirá sucediendo en las dos siguientes, el único pero es la mezcla de atributos, algo intrínseco al Spectrum, y que no es inconveniente para nuestra placentara aventura.

Una melodía machacona y pegadiza (en algún momento nos sorprenderemos silbándola) nos acompaña en el menú antes de comenzar el juego. Menú en el que podemos elegir entre el teclado y joysticks Kempston o Sinclair para el manejo del personaje. Como curiosidad, en la cinta de cassette del juego viene la canción grabada por Mike Berry, cantante muy conocido a la hora de comer en su casa. El resto de sonidos son los habituales efectos caminando, al coger objetos, rozar enemigos... correctos.



Wilma frente a la oficina de correos

Muchas cosas nos dejamos en el tintero de esta maravilla de juego con el que se despedía de la saga su programador, Chris Hinsley. Una despedida gloriosa. Un programa lleno de detalles y de situaciones divertidas (ya encontrarás al tiburón y te tocará machacar teclas para escapar). Necesitaríamos un monográfico sólo para hablar de él, pero mejor que descubráis vosotros mismos todos los recovecos del juego incluido su huevo de pascua en forma de clon de "Asteroids".

Si te gustan las aventuras y no has jugado al Everyone's A Wally estás tardando en cargarlo en tu Spectrum o emulador preferido, no te arrepentirás. Este juego es la cúspide de la saga, aunque luego vendrían otras dos entregas más.

Valoraciones

Originalidad:	[10]	
Gráficos:	[10]	
Sonido:	[7]	
Jugabilidad:	[10]	
Adicción:	[10]	
Dificultad:	[10]	

Trucos:

Puedes encontrarlos en The Tip Shop

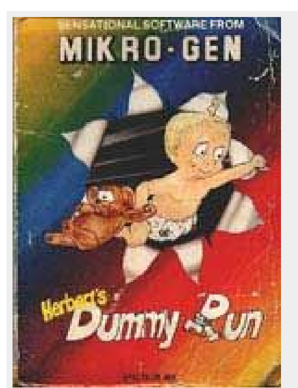
http://www.the-tipshop.co.uk/cgi-bin/search.pl?name=Everyone's+A+Wally

Descárgalo de:

- WOS

http://www.worldofspectrum.org/infoseek.cgi?regexp=^Everyone's+A+Wally$&pub=^Mikro-Gen+Ltd$

Herbert's Dummy Run



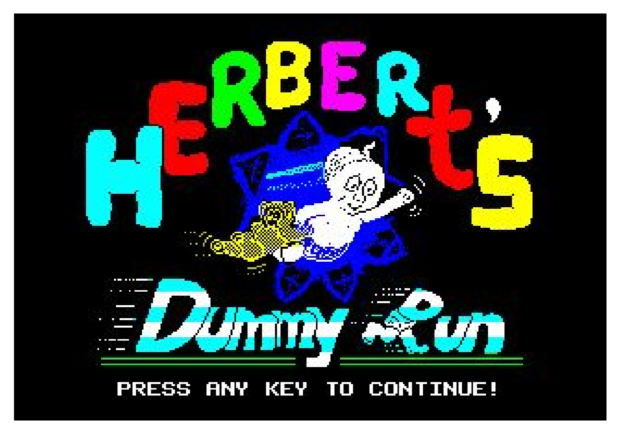
Título	Herbert's Dummy Run
Género	Videoaventura
Año	1985
Máquina	48K
Jugadores	1 Jugador
Compañía	Mikro-Gen
Autor	David J. Perry

Herbert, ese niño revoltoso que en la anterior entrega de la serie hacía que bajara nuestro nivel de energia, ha crecido. Ahora es capaz de caminar de pie él solito, pero sigue siendo un trasto. Se encuentra perdido en unos grandes almacenes y debe encontrar el camino de vuelta a casa lo antes posible. Para ello hay que dar uso a la infinidad de objetos que, como viene siendo habitual en este tipo de juegos, están diseminados por las diferentes estancias.

El primer juego de la serie programado por David Perry si destaca en algo es por su falta de originalidad. Básicamente es muy similar a Pyjamarama, si bien cambia el personaje y el entorno. Todo el desarrollo del juego sucede en el interior de unos grandes almacenes, cosa que, después de ver el derroche de imaginación de su precuela Everyone's, le resta muchos enteros. El juego no es malo, ni mucho menos, tiene el nivel que caracteriza a la serie, y para los fans va a dar otro rato de divertimento. Pero quizá ése es el problema, que es un poco más de lo mismo y sin ninguna mejora aparente, como sucedía hasta ahora, en el que cada entrega mejoraba a su predecesora.

Poco hay que comentar sobre este Herbert's Dummy Run, si acaso la serie de sub-juegos a los que tendremos que enfrentarnos para ir avanzando en nuestra aventura, juegos con los que podemos entretenernos independientemente del desarrollo de la misma, sólo por pasar el rato.

Los gráficos del juego son muy buenos, coloridos, con gran tamaño y que nos hacen sumergirnos en el ambiente en que está nuestro personaje. Herbert, estando bien diseñado, tiene unos andares un tanto extraños, como si tuviera las piernas cortas en relación al resto del cuerpo, pero su manejo es bueno y reacciona tal y como esperamos al pulsar las teclas de control o dirigirlo con el joystick que, como es habitual, puede ser de protocolo Sinclair o Kempston.



Pantalla inicial



Disparando chupetes y esquivando dados

El sonido es correcto, ligeramente más flojo (especialmente la música del menú) que en las anteriores entregas.

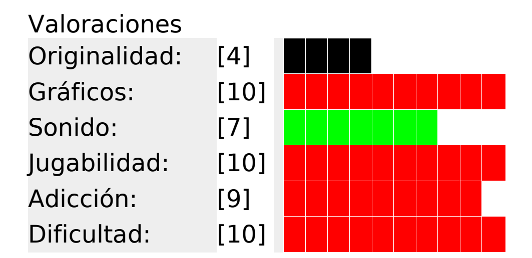


Sección HI-FI

Resumiendo, da la impresión de que cogieron el motor del Pyjamarama, le cambiaron los gráficos, la música y el argumento, pero poco más. Es el juego de la serie que menos me ha gustado, siempre visto en el conjunto de las cinco entregas incluyendo el Automania, que no era una videoaventura y que, quizá, si en lugar de englobarlo en la serie Wally y tener un personaje afín lo hubieran lanzado como un juego independiente, las conclusiones serían diferentes.

Aunque parezca una contradicción con todo lo escrito anteriormente, el juego es bastante bueno, pero falla en su falta de aportación de novedades a la saga.

Valoraciones



Originalidad:	[4]
Gráficos:	[10]
Sonido:	[7]
Jugabilidad:	[10]
Adicción:	[9]
Dificultad:	[10]

Trucos:

Puedes encontrarlos en The Tip Shop

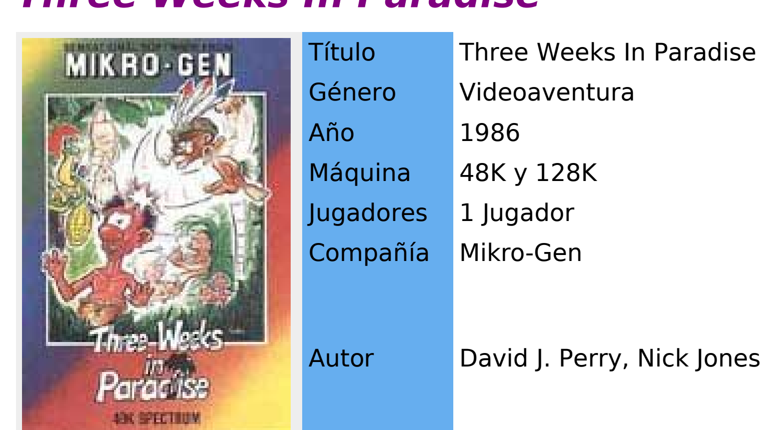
http://www.the-tipshop.co.uk/cgi-bin/search.pl?name=Herbert's+Dummy+Run

Descárgalo de:

- WOS

http://www.worldofspectrum.org/infoseek.cgi?regexp=^Herbert's+Dummy+Run$&pub=^Mikro-Gen+Ltd$

Three Weeks In Paradise



Título	Three Weeks In Paradise
Género	Videoaventura
Año	1986
Máquina	48K y 128K
Jugadores	1 Jugador
Compañía	Mikro-Gen
Autor	David J. Perry, Nick Jones

Después del bache de la anterior entrega, en opinión (muy subjetiva) de quien esto escribe, cerraron la saga de la mejor manera que podían hacerlo, con un juego muy digno de la familia Week: Three Weeks In Paradise.

Wally y su familia se han embarcado en un crucero de placer para descansar de sus últimas peripecias, pero por un desgraciado accidente se han visto obligados a desembarcar en una desconocida isla. Lejos de estar deshabitada, los nativos del selvático peñote en medio del océano no son nada amigables. Han capturado a nuestra querida Wilma y la tienen colgada de un árbol, cabeza abajo, cual jamón. Y con nuestro travieso hijo quieren hacer una sopa. Con esta misión, perdidos en la selva, y con nuestra familia en grave peligro, comenzamos la aventura. El objetivo está claro: salvarlos.



Divertida pantalla de carga

Si bien no se puede decir que Three Weeks In Paradise sea un derroche de originalidad, si es lo que nos podíamos imaginar como continuación y final de la saga. No supera a Everyone's a Wally con novedades sustanciosas, pero desde luego es un juego muy entretenido.



Jungle fever

El grafista derrochó no sólo imaginación diseñando gráficos y pantallas, también las llenó de detalles divertidos que nos harán dibujar una sonrisa con bastante frecuencia. Partiendo del mismo personaje, Wally, con un pañuelo a modo de gorro con un nudo en cada esquina y su "taparrabos" a lo Tarzán, pasando por esas señales indicando la dirección a seguir en medio de la jungla e infinidad de ojos que nos miran entre los árboles, y muchas cosas más que iréis viendo en el devenir de vuestros paseos. Las pantallas, numerosas y coloridas, son de un diseño muy bonito, con playas, jungla, fondos de mar con sus caballitos, etc. Como opción podemos cambiar el modo en el que se muestra el color de Wally en el juego, entre el amarillo habitual que crea la típica mezcla de color con los atributos y otro modo en el que Wally se adapta al color del escenario. Elección al gusto de cada cual.

El movimiento de Wally es el habitual, bien logrado y con una manejabilidad alta. En esta ocasión una novedad ha sido añadida: para coger los objetos no bastará con pasar por encima de ellos, ahora tendremos que pulsar la tecla 1 ó 2 dependiendo del "hueco" donde queramos colocar el objeto.

La música da un salto de calidad bastante alto respecto a las entregas anteriores. La melodía del menú está bien realizada y, como otra novedad más, se incluye música durante el juego. Esta música es diferente a la del menú, y también es pegadiza, aunque puede llegar a ser molesta por lo repetitiva. Afortunadamente también tenemos opción de desactivarla durante el juego.

Three Weeks In Paradise salió en dos versiones: para ordenadores de 48K y 128K. Las diferencias son apreciables y aprovecharon los avances que ofrecía la máquina superior. Para empezar las músicas y sonidos están creados en modo 128K utilizando el chip de sonido incorporado. Y continuando con el número de pantallas, ya que en el modelo con más capacidad fueron aumentadas respecto a la versión 48K, al igual que los objetos que utilizaremos para llevar a buen término nuestra misión.



Nadando con gran estilo

Las conclusiones no pueden ser más positivas. Three Weeks In Paradise es un gran juego, aprovechando al límite el ordenador de 48K y sacando a la luz las virtudes del 128K. La pena es que con él terminase una saga mítica que dio muchos momentos de diversión a los amantes de las videoaventuras, pero nos queda el consuelo de su despedida por la puerta grande. ¿Se habrá jubilado Wally?...

Valoraciones

Originalidad:	[7]
Gráficos:	[10]
Sonido:	[9]
Jugabilidad:	[10]
Adicción:	[10]
Dificultad:	[10]

Trucos:

Puedes encontrarlos en The Tip Shop

http://www.the-tipshop.co.uk/cgi-bin/search.pl?name=Three+Weeks+In+Paradise

Descárgalo de:

- WOS

http://www.worldofspectrum.org/infoseek.cgi?regexp=^Three+Weeks+In+Paradise$&pub=^Mikro-Gen+Ltd$

LINKS

- Chris Hinsley: http://www.worldofspectrum.org/infoseekpub.cgi?regexp=^Chris+Hinsley$
- David J. Perry: http://www.worldofspectrum.org/infoseekpub.cgi?regexp=^David+J%2e+Perry$
- Nick Jones: http://www.worldofspectrum.org/infoseekpub.cgi?regexp=^Nick+Jones$
- Mikro-gen: http://www.worldofspectrum.org/infoseekpub.cgi?regexp=^Mikro%2dGen+Ltd$
- WOS: http://www.worldofspectrum.org
- Web personal de David J. Perry: http://www.dperry.com/



SIEW

Espectrum

Lo que para mucha gente está claro es que los ordenadores Sinclair, sobre todo el Spectrum, tuvieron un gran impacto en nuestro país. Para la gente está claro que existen muchas páginas en español que tratan el tema. Pero... ¿y una página, no solo escrita por un español, sino que además trate en profundidad el tema del Spectrum en España, más allá de la simple recolección de archivos TAP de juegos de su colección personal o más alla de una simple muestra de las máquinas que posee? Esa es la página de Horace. Esa es la página Espectrum.

Es ésta una página muy consultada (para comprobarlo no hay más que acudir al libro de visitas) incluso por extranjeros, que trata de aglomerar información sobre el Spectrum en nuestro país, teniendo alguna sección que de por sí justifica el visitarla. Su autor es Horace, quizás conocido por muchos por su participación intermitente en es.comp.sistemas.sinclair.

¿Qué nos ofrece Horace en Espectrum?

DISEÑO

Una vez más, nos encontramos ante un diseño espartano y un tipo de interactividad estándar, basado en un frame izquiero que nos da paso a cada una de las secciones de la página. Este diseño tan sencillo hace que quien no tenga el navegador más extendido en el planeta Tierra (como por ejemplo Konqueror, que es el que utiliza el autor de este comentario) pueda acceder sin problemas a sus contenidos. En ningún momento aparece el típico mensaje de cuál es la resolución o el navegador recomendados.

El frame central se basa en un agradable color blanco de fondo y unas fuentes en diversos colores, según la sección. Esto en algún momento puede ser chocante, pero el efecto nunca es desagradable como, por ejemplo, si se hubieran utilizado otros colores de fondo. Hay varías imágenes por ahí repartidas, pero nunca hacen de la navegación una experiencia pesada. El frame izquierdo tiene un diseño

agradable. Se fundamenta en un menú que nos permite navegar entre las distintas secciones de la página, con la apariencia del menú que podemos hallar al encender algunos modelos de Spectrum. De hecho, al pasar el ratón por algún título de alguna de las secciones veremos como cambia de color a azul claro (¡como en el Spectrum!). Después vemos un contador de visitas desde el 98 exageradamente grande, y por último una nota de copyright, en la que se utiliza un gracioso sprite de Horace, el protagonista de tres aventuras en Spectrum.



Cada día, un error de Spectrum distinto en la portada de esta web

La interactividad es la típica de las páginas con este diseño. Al pulsar sobre el nombre de alguna de las secciones en el frame izquierdo nos aparecerá su contenido en el frame central. En el caso de que una sección contenga subsecciones, estás podrán ser seleccionadas en el propio frame central. En ningún momento, al pulsar sobre alguna de las secciones (excepto en la consabida sección de enlaces), se nos abrirá una nueva ventana, lo cual suele ser bastante molesto.

CONTENIDO

Ya desde un primer momento la página destila un olor a español. En la página principal, podemos ver el nombre de la web sobre un fondo de pantallas de carga de juegos españoles (el Humphrey, el Game Over, el Army Moves, el Sir Fred, etc.). En esta misma página principal encontramos un pequeño texto de introducción que nos dice el por qué de la creación de esa página. También podemos acceder desde aquí al libro de visitas, tanto para dejar nuestro comentario como para sorprendernos ante la cantidad de gente que ya lo ha hecho. Un poco más abajo una serie de enlaces nos permiten movernos por el anillo de webs Espectrum (al que pertenece también esta página) y, por último, una serie de enlaces a webs recomendadas, sobre todo relacionadas con la emulación. Pasemos a describir qué nos ofrecen el resto de secciones.

La primera de todas es la omnipresente sección de novedades (decimos omnipresente porque esta sección suele estar presente en la práctica totalidad de las páginas web). Lo que vemos ahí es que, aunque no se actualiza de forma regular (es decir, cada semana, cada mes, etc., de forma matemática), esta página si que recibe actualizaciones en espacios cortos de tiempo. En el momento de escribir esto, la última actualización data del 6 de Junio del 2004, así que parece ser que tenemos Espectrum para rato. La sección del archivo, dedicada al almacenamiento de diversos programas, se divide a su vez en una serie de subsecciones; podremos descargar juegos y programas de Spectrum realizados en nuestro país, o al menos traducidos a nuestro idioma (se ofrece también un buscador para encontrar un juego específico), podremos obtener emuladores de Spectrum para diversos sistemas operativos que han sido realizados en nuestro país (una subsección muy bien elaborada, porque podremos buscar por sistema operativo), podremos acceder a una serie de documentos sobre los más variopintos temas (algo escasos en número) y, por último, descargar utilidades relacionadas con el Spectrum desarrolladas en España (muy completita). Además, dentro de esta sección de Archivo, se nos muestra un listado con los 10 programas más descargados de la web (indicando si se trata de un emulador, un juego, etc.) y los 10 enlaces más visitados desde la página (enlaces externos; el ganador indiscutible es World Of Spectum). Es curioso como de los 10 programas más descargados, 9 de ellos son emuladores. Para obtener juegos la gente parece ser que prefiere acudir a otras fuentes.



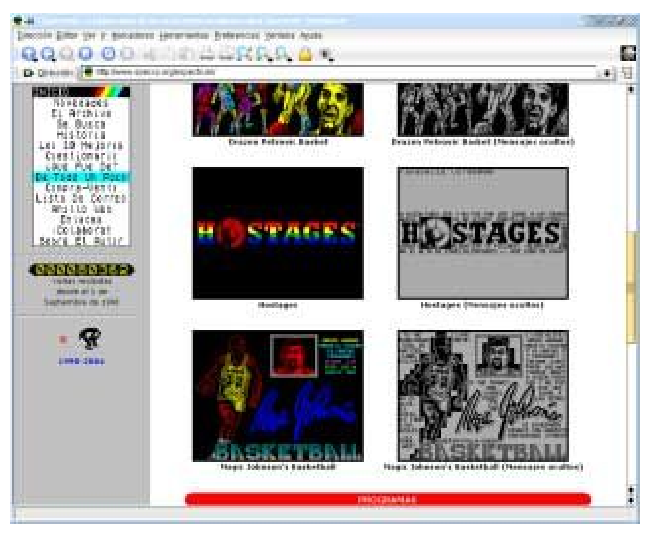
La Abadía del Crimen arrasa en casi todas las categorías de Los 10 Mejores

En la sección de Se Busca se indica cuáles son los títulos que, según el autor, quedan por obtener para completar esta antología de juegos españoles. La sección de historia es muy interesante. Desglosado por casas creadoras de software, podremos acceder a entrevistas a diversos programadores y reportajes sobre estas casas que aparecieron en diversos medios (sobre todo Microhobby y Micromania). Todos estos documentos están transcritos a html, por lo que el acceso a los mismos es rápido y se nos permite hacer operaciones con el texto cómodamente.

La sección de Los 10 Mejores deja espacio a la

subjetividad, aunque en este caso no se está expresando la opinión de Horace, sino la de los visitantes de la página, que mediante un formulario pueden votar a sus juegos preferidos en diverso sapartados, como originalidad, música, etc. Quizás, un inconveniente que encontramos en esta sección, es que no haya una posibilidad de descarga directa, como por ejemplo había en la web de El Mundo del Spectrum, que descansa en el sueño de los justos. Nos referimos a que si vemos, por ejemplo, que el juego en la primera posición de la lista de los más originales es la Abadía del Crimen, estaría bien poder pinchar sobre el nombre del juego en el mismo listado y descargarlo, en lugar de tener que visitar otra sección distinta para ello.

La sección de Cuestionario es bastante curiosa. Tras introducir nuestro nombre y dirección de correo en un formulario, podremos seleccionar el número de preguntas que se nos formularán (de 5 a 25, en intervalos de 5). Desgraciadamente, al cuestionario en sí mismo no se pudo acceder desde Konqueror y se tuvo que utilizar Mozilla (desde Linux). Se contabiliza tanto el acierto en las respuestas como el tiempo empleado. Si queremos compararnos con el resto de la gente que ha realizado este curioso ejercicio, podremos consultar un listado con la totalidad de participantes, o un ranking con los 10 mejores.



Menudos mensajes ponía la gente en las pantallas de carga

La sección que viene a continuación es, sin duda, la preferida de la persona que está haciendo este comentario: ¿Qué fue de?. Es de gran interés porque se nos muestra cuál ha sido la trayectoria de diversos profesionales y compañías en el mundo del Spectrum hasta la actualidad. Podremos enterarnos de datos tan curiosos como que "Gominolas", el compositor de las músicas de muchos juegos de Topo, actualmente se encuentra en Pyro Studios y ha participado en la elaboración del juego Commandos, o como que "Charly" Granados en la actualidad sigue en Zigurat, que ha abandonado el mundo de los juegos de ordenador para dedicarse a las recreativas. Sin dudas, una de las secciones MÁS INTERESANTES (con mayúsculas) que he podido consultar en mi vida. Podemos incluso ver fotografías de estas personas, actuales o no, para hacernos una idea de cómo son sus caras. Para que no tengamos duda de la veracidad o de la actualidad de la información, se incluye la fuente de dónde se ha obtenido y en qué fecha.

La sección de De todo un Poco es también muy interesante. Es una especie de miscelánea donde agrupar el contenido que no encaja en ninguna de las otras secciones. Podremos enterarnos de unos pocos bugs en juegos españoles, podremos sorprendernos con la cantidad de mensajes ocultos que dejaban los programadores en las pantallas de carga, y algunas pocas cositas más.

La sección de Compra-Venta no está en absoluto mal, pero no estaría de más que se incluyera la fecha de cuándo fue publicado el anuncio, para que se sepa cómo de actual es el mismo, y evitar preguntar por algún producto al vendedor que seguramente ya habrá vendido u ofrecer un producto a un comprador que seguramente ya habrá adquirido.

Las siguientes secciones son las típicas de enlaces, petición de colaboraciones, anillo web, etc. Por último, si todo lo anterior nos parece insuficiente y nos aburrimos, siempre podremos leer qué es que lo que nos tiene que contar Horace sobre sí mismo en la última sección: Sobre el Autor.

CONCLUSIONES

Una página que va más allá de la mera recolección de programas o el simple almacenamiento de textos, y nos ofrece contenido muy interesante sobre la historia del computador Spectrum en nuestro país, y los protagonistas de dicha historia. Aunque el diseño es algo flojo, sin nunca llegar a ser desagradable ni malo, el contenido justifica claramente su visita. También dispone de secciones interactivas para el caso de que estemos un poco aburridos o queramos participar (el cuestionario, votar por los juegos, etc...).

Como cosas que mejorar; las pocas que se han comentado anteriormente: incluir la fecha de los anuncios de compra-venta, la posibilidad de descarga directa desde los listados de Los 10 Mejores, y un diseño más elaborado. Poco más. Es una web muy recomendable.

PUNTUACION: 8

LINKS

- *Espectrum : http://www.speccy.org/espectrum/*



SIEW

CREANDO UNA AVENTURA CONVERSACIONAL CON Z88DK (II)

Seguimos en esta ocasión por donde lo dejamos en la entrega anterior. Recordemos que el objetivo de esta serie de artículos es practicar con las funciones de texto de la librería z88dk, por medio de un ejemplo de programa que, aun siendo posible que se desarrolle con otras herramientas, como BASIC, parsers, etc., es bastante ilustrativo, y además nos permite estar haciendo un juego desde el primer momento.

En la entrega anterior habíamos comenzado a escribir las aventuras de Guybrush Threpwood, un aspirante a pirata, que debía dirigirse a un maestro en la taberna de la isla de Melêe e impresionarle con algún truco. Habíamos dado los pasos precisos para que nuestro personaje fuera capaz de desplazarse entre las distintas habitaciones de la taberna, con las funciones de texto del z88dk, que habíamos visto que eran muy parecidas a las del C estándar de PC. También vimos como aplicar unos efectos interesantes al texto (negrita, cursiva, subrayado y texto inverso).

En el presente artículo veremos alguna técnica para incorporar objetos a nuestro mundo, incluyendo la posibilidad de interactuar con ellos, y la creación de un inventario. También veremos como aplicar efectos de color al texto (hasta ahora, todo era en un aburrido blanco y negro). Sin más dilación comenzamos.

¿Qué necesitamos?

Si seguimos la entrega anterior no tenemos que realizar ninguna preparación adicional. Todo el código que vayamos añadiendo se incluirá en los archivos aventura.c y datos.h.

Creando los objetos

Lo primero de todo es crear la estructura de datos que nos permitirá que los objetos puedan ser manejados durante el juego, igual que se hizo en el caso de las habitaciones. Esta estructura de datos para los objetos contendrá una serie de valores que serán de utilidad, como el nombre del objeto, dónde se encuentra, cuánto pesa, etc.

En concreto, podemos añadir el siguiente código a datos.h (da igual si es antes o después de la definición de THabitacion):

```
typedef struct
{
        char nombre[35];
        int localizacion;
        int peso;
} TObjeto;
```

Todos los objetos de nuestro juego serán variables del tipo TObjeto. El primer campo (nombre) contendrá el nombre con el que aparecerá el objeto en la pantalla, como por ejemplo: "una antorcha", "un anillo", "la espada", etc. El campo de localizacion indicará en que posición del mapeado de juego se encuentra el objeto, aunque también puede indicar diferentes estados del objeto (como por ejemplo, que el objeto está en nuetros inventario, que está destruido, que todavía no se ha creado, que lo llevamos puesto, etc.). Por último, el peso, como es evidente, indica cuánto pesa el objeto. Es corriente en las aventuras que el protagonista sólo pueda llevar un número limitado de cosas, y el campo peso permite controlar esto.

Hablemos del campo localización. Hemos dicho que ese campo nos va a indicar en qué lugar se encuentra el objeto. Si recordamos, cada habitación de nuestra aventura tenía un identificador, que coincidía con su posición en el array de habitaciones (menos uno). Si queremos que el objeto esté en una de las habitaciones, el campo localización deberá contener el identificador de esa habitación (o la posición en el array de habitaciones). Por ejemplo, supongamos que queremos crear un objeto, una espada, que se encuentre en la habitación número 5 (la cocina). Lo que debemos hacer es almacenar en el campo localización de la variable que represente a la espada el valor 5 (más adelante veremos ejemplos).

El campo de localización es un campo dinámico; es decir, puede cambiar de valor durante el transcurso de la aventura. Por ejemplo, si la espada deja de

estar en la cocina para pasar a estar en el salón (porque el personaje la haya cogido de la cocina y la haya dejado en el salón), el valor de localización pasará a ser 4.

Por lo tanto, el campo localización indica la habitación en la que se encuetra el objeto. Sin embargo, existe una serie de localizaciones especiales, que no se corresponden con habitaciones de nuestra aventura. Uno de estos valores es el -1. Si el campo localización de un objeto vale -1, significa que el objeto está en posesión del jugador (y que por lo tanto, al consultar el inventario o listado de objetos que porta, se le indicará que lo lleva). Aparte podemos inventarnos todas las localizaciones especiales que quisieramos, por ejemplo, la 254 para los objetos que tenemos puestos, la 255 para los que están destruidos, etc. (más adelante hablaremos también de ello).

Antes de seguir programando, hemos de pensar: ¿qué objetos va a tener nuestra aventura? En nuestro caso concreto vamos a tener tres:

- Una espada, que encontraremos en la cocina de la taberna.
- Una jarra, que portará el jugador consigo desde el comienzo de la aventura.
- Una antorcha, que podremos recoger en el callejón.

De momento vamos a incluir la espada y la jarra en nuestra aventura. La antorcha es un tipo de objeto más complicado que trataremos más adelante. En el archivo aventura.c, en el método main, justo después de la inicialización de las habitaciones, podemos incluir el siguiente código (el código en rojo es el que se añade):

```
THabitacion habitaciones[6];
TObjeto objetos[4];
inicializarHabitaciones(habitaciones);
inicializarObjetos(objetos);
```

Como en el caso de las habitaciones, creamos un array que contendrá todos los objetos de nuestra aventura. En nuestro caso, este array tendrá ta-maño 4, para la espada, la jarra, y dos para la antorcha (no os impacientéis, luego entenderéis por qué se usan dos objetos para la antorcha). Evidentemente, si ahora compilamos no va a funcionar, porque falta por crear la función inicializarObjetos que, como en el caso de inicializarHabitaciones, se encarga de rellenar el array de objetos. Nuestra función inicializarObjetos podría tener la siguiente forma (se debe recordar que tiene que estar antes de la función main):

```
void inicializarObjetos(TObjeto objetos[])
{
        strcpy(objetos[0].nombre,"una espada");
        objetos[0].localizacion = 5;
        objetos[0].peso = 5;

        strcpy(objetos[1].nombre,"una jarra");
        objetos[1].localizacion = -1;
        objetos[1].peso = 3;
}
```

(la antorcha todavía no la hemos creado... sí, somos pesados con la antorcha). El primer objeto es la espada, y al principio de la aventura se encontrará en la habitación cuyo identificador sea el 5 (la cocina); decimos al principio porque, como hemos comentado antes, eso puede cambiar. La espada tiene un peso de 5. El siguiente objeto es la jarra, cuya localización inicial es la -1, que hemos dicho que era la localización especial que usábamos para indicar que el objeto estaba en posesión del protagonista de la aventura.

El último paso es que, junto a la descripción de las habitaciones, se nos indiquen los objetos que podemos encontrar en ellas. Eso, evidentemente, es necesario hacerlo en la función escribirDescripcion, y se hará de forma similar a como se escribían las posibles direcciones; lo único es que tendremos que recorrer el array de objetos comprobando cuáles están en la habitación. Por lo tanto, tenemos que pasar como parámetro el array de objetos. La función podría quedar de la siguiente forma (el código en rojo es nuevo):

```
void escribirDescripcion(THabitacion habitaciones, int habitacion, TObjeto objetos)
{
        int hayObjetos = 0;
        int i;

        printf(habitaciones[habitacion].descripcion);
        printf("\n\n");
        printf("Salidas:");
        if (habitaciones[habitacion].direcciones[0] != 0)
                printf(" %c[4mNorte%c[24m",27,27);
```

```
        if (habitaciones[habitacion].direcciones[1] != 0)
                printf(" %c[4mEste%c[24m",27,27);
        if (habitaciones[habitacion].direcciones[2] != 0)
                printf(" %c[4mSur%c[24m",27,27);
        if (habitaciones[habitacion].direcciones[3] != 0)
                printf(" %c[4mOeste%c[24m",27,27);
        printf("\n\n");
       printf("En la habitacion puedes ver:");
        for (i=0;i<4;i++)
                if (objetos[i].localizacion == habitaciones[habitacion].id)
                {
                        printf(" %s",objetos[i].nombre);
                        hayObjetos = 1;
                }
        if (hayObjetos == 0)
                printf(" nada");
        printf("\n\n");
}
```

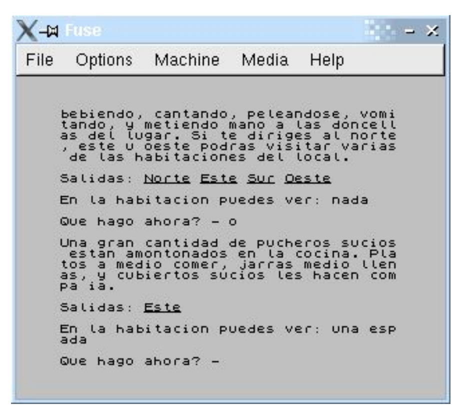
Como hemos añadido un parámetro de entrada más, en todas las líneas a las que se llame a la función se debe añadir también ese parámetro. Estas líneas (en la función main) quedarían de la siguiente forma:

```
escribirDescripcion(habitaciones,habitacion,objetos)
```

El código que hemos añadido recorre el array de objetos comprobando para cada uno si su localización corresponde con el identificador de la habitación actual. En el caso de que sea así escribe su nombre (fíjate en el espacio en blanco antes del %s). La variable hayObjetos nos sirve para determinar si hay algún objeto en la habitación. En un principio vale 0, y si la localización de algún objeto se corresponde con el identificador de la habitación actual, valdrá 1. Solo en el caso de que valga 0 se mostrará el mensaje "nada". Ahora ya podemos compilar y ejecutar la aventura, dirigiéndonos hacia la cocina para comprobar que la espada está allí. No podemos saber nada de la jarra, porque todavía no hemos implementado el inventario, lo haremos en la siguiente sección.

Coger y dejar objetos. El inventario

A partir de este momento, el secreto para poder realizar acciones con los objetos durante la aventura consiste nada más que en añadir posibles frases a usar por el jugador en el intérprete de comandos. Si recordamos, el intérprete de comandos era la parte del código que leía la cadena de texto introducida por el jugador y comparaba con el vocabulario conocido con el programa. Para incluir los comandos de inventario y coger y dejar objetos debemos añadir código a esta parte.



¿Quién se habrá dejado esta espada aquí tirada en la cocina?

Implementemos primero el inventario. Cuando el jugador teclee "inventario" o "i", se deberán mostrar por pantalla todos los objetos que porta. Podemos añadir el siguiente código al intérprete de comandos, dentro del método main (el código en rojo es el que se ha añadido):

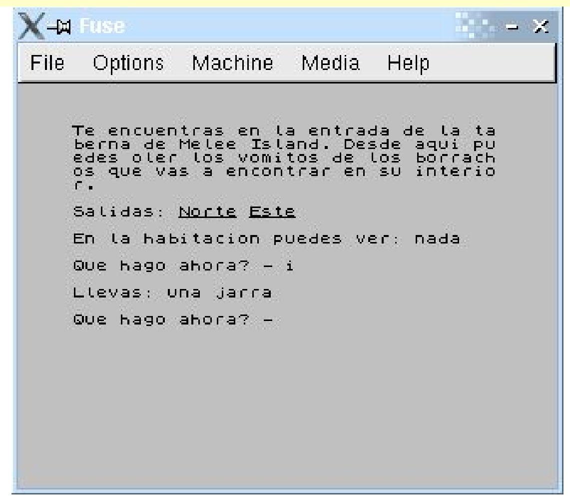
```
                }
                else
                        printf("\n\nNo puedo ir en esa direccion\n\n");
        }
        else if (strcmp(comando,"i") == 0 || strcmp(comando,"inventario") == 0)
        {
                hayObjetos = 0;
                printf("\n\nLlevas:");
                for (i = 0; i<4;i++)
                        if (objetos[i].localizacion == -1)
                        {
                                printf(" %s",objetos[i].nombre);
                                hayObjetos = 1;
```

```
                }
            if (hayObjetos == 0)
                printf(" nada");
            printf("\n\n");
    }

    else
            printf("\n\nNo entiendo lo que dices\n\n");

}
```

La variable hayObjetos se debe definir al comienzo de la función main y cumple el mismo objetivo que en la función describirHabitacion. Lo único que se hace en el caso del inventario es recorrer el array de objetos escribiendo el nombre de aquéllos para los que el valor de localización sea -1, o "nada" en el caso de que no haya ninguno que cumpla esta condición. Si compilamos y ejecutamos, por fin veremos la jarra en nuestro juego al teclear "i" o "inventario".



Esperemos que esta jarra nos sirva de algo en nuestra aventura

Para que el jugador pueda coger y dejar objetos, lo único que tenemos que hacer, nuevamente, es añadir los comandos adecuados al intérprete de comandos. La implementación es tan simple como hacer que cuando cojamos un objeto, la localización del mismo pase a ser -1, y que cuando dejemos un objeto, la localización del mismo pase de ser -1 a ser el identificador de la habitación actual. Veamos primero cómo implementar la parte de coger objetos (el código en rojo es código añadido):

```
else if (strcmp(comando,"i") == 0 || strcmp(comando,"inventario") == 0)
{
        hayObjetos = 0;
        printf("\n\nLlevas:");
        for (i = 0; i<4;i++)
                if (objetos[i].localizacion == -1)
                {
                        printf(" %s",objetos[i].nombre);
                        hayObjetos = 1;
                }
        if (hayObjetos == 0)
                printf(" nada");
        printf("\n\n");
}
else
// Comandos con más de una palabra
{
        strcpy(palabra,strtok(comando," "));
        if (strcmp(comando,"coger") == 0)
        {
                strcpy(palabra,strtok(0,"\0"));
                if (palabra == 0)
                                printf("\n\nNecesito que me digas que tengo que
coger\n\n");
                else
                {
                        hayObjetos = 0;
                        i = 0;
                        while (hayObjetos == 0 && i<4)
                        {
                                if (strcmp(objetos[i].nombre,palabra) == 0 &&
```

```
                                        objetos[i].localizacion == habitacion+1)
                                {
                                        objetos[i].localizacion = -1;
                                        hayObjetos = 1;
                                        printf("\n\nHe cogido %s\n\n",palabra);
                                }
                                i++;
                        }
                        if (hayObjetos == 0)
                                printf("\n\nNo puedo hacer eso\n\n");
                }
        }
        else
                printf("\n\nNo entiendo lo que dices\n\n");
}
```

Para que este código funcione, se debe crear una nueva variable (al inicio de la función main):

```
char palabra[50];
```

¿Qué significa todo lo que hemos añadido? En primer lugar, se ha realizado una modificación importante al intérprete de comandos. hasta ahora, solo era capaz de interpretar comandos de una única palabra. Ahora le hemos añadido la posibilidad de comprender comandos de más de una palabra. Simplemente, primero comprobamos si el comando introducido por el jugador se corresponde con alguno de los de una palabra. y al final introducimos el código que interpreta más de una. Este código empieza con la instrucción strcpy (palabra,strtok(comando," ")).

La función strtok es otra de las que nos ofrece z88dk y su uso es exactamente igual al del estándar en el PC; strtok(cadena1,cadena2) devuelve la primera subcadena de cadena1, estando todas las subcadenas de cadena1 delimitadas por el carácter especificado en cadena2. Así pues, si cadena2 vale " " (espacio en blanco) como en el código anterior, la llamada a esta función devolverá los primeros carácteres de cadena1 hasta llegar al primer espacio en blanco (o el final de la cadena), es decir, la primera palabra. A partir de este momento, si vamos llamando a strtok pasando el valor 0 como cadena1, se nos irán devolviendo subcadenas sucesivas a partir de la cadena inicial. Hay que destacar que esta cadena inicial queda modificada; además, el código anterior, al compilarlo, nos mostrará una serie de warnings, que pueden ser ignorados (el programa funciona correctamente). Como lo que se devuelve es una cadena, es necesario utilizar strcpy para almacenar el valor devuelto en una variable de tipo char[].

Si la primera palabra es "coger", entramos en la parte del código que interpreta este comando. Se utiliza de nuevo strtok para almacenar en palabra el resto del comando (pues se indica como delimitador el símbolo de final de cadena \0). Si el comando solo se compone de la palabra coger se mostrará un mensaje de error adecuado. Lo siguiente es inicializar las variables i, que se utilizará como contador para recorrer el array de objetos, y hayObjetos, que en este caso se utilizará para saber si se ha encontrado un objeto con el mismo nombre que el tecleado por el jugador. A continuación, se recorre el array de objetos buscando un objeto cuyo nombre se corresponda con el introducido por el jugador. El principal inconveniente es que el jugador tiene que introducir el nombre completo, incluyendo el artículo, en el caso de que se hubiera puesto. Por ejemplo, si el nombre del objeto es "una espada", el jugador deberá teclear "coger una espada" para poder obtenerla. Se deja como ejercicio al lector arreglar este pequeño fallo.

Al recorrer el array no sólo se comprueba si existe algún objeto cuyo nombre se corresponda con el introducido, sino que también el objeto debe encontrarse en la misma habitación que el jugador. Coger el objeto consiste nada más en cambiar el valor del campo localización del objeto a -1.

Obsérvese que, tanto si el objeto existe como si no, tanto si el objeto está en la misma habitación como si no, se muestra el mismo mensaje de error ("No puedo hacer eso"). Esto es básico para no darle pistas al jugador sobre los objetos que existen en nuestro juego.

El código para dejar objetos es prácticamente igual; la única diferencia es que comprobamos que para dejar un objeto éste se encuentre en el inventario (localizacion = -1) y que dejar un objeto significa cambiar el valor del campo localizacion al del identificador de la habitación actual (el código en rojo es el que se ha añadido):

```
                        if (hayObjetos == 0)
                                printf("\n\nNo puedo hacer eso\n\n");
                }
        }
        else if (strcmp(comando,"dejar") == 0)
        {
                strcpy(palabra,strtok(0,"\0"));
```

```
		if (palabra == 0)
				printf("\n\nNecesito que me digas que tengo que dejar\n\n");
		else
		{
			hayObjetos = 0;
			i = 0;
			while (hayObjetos == 0 && i<4)
			{
				if (strcmp(objetos[i].nombre,palabra) == 0
					&& objetos[i].localizacion == -1)
				{
					objetos[i].localizacion = habitacion+1;
					hayObjetos = 1;
					printf("\n\nHe dejado %s\n\n",palabra);
				}
				i++;
			}
			if (hayObjetos == 0)
			    printf("\n\nNo puedo hacer eso\n\n");
		}
	}
	else
		printf("\n\nNo entiendo lo que dices\n\n");
}
char palabra[50];
```

Un último detalle que nos queda por comentar es el del peso de los objetos. Recordemos que uno de los campos de TObjeto era el peso. Supongamos que el peso máximo que puede llevar un jugador es de 6. Lo que tenemos que añadir para poder controlar el peso en nuestro juego es lo siguiente:

- Crear una variable que almacene el peso de los objetos portados por el jugador.
- Controlar que el peso total al coger un objeto no supere el peso máximo que puede llevar un jugador. En caso de no ser así sumar el peso del objeto recogido al peso total.
- Al dejar un objeto, restar su peso al peso total.

Para resolver el primer punto, creamos la variable pesoTransportado, al inicio de la función main:

```
	int pesoTransportado;
```

Y la inicializamos con el peso del objeto que porta el jugador nada más empezar. Esto lo podemos hacer justo después de inicializar los objetos:

```
inicializarHabitaciones(habitaciones);
inicializarObjetos(objetos);
pesoTransportado = objetos[1].peso;
```

Para resolver el segundo punto, añadimos el siguiente código a la hora de manejar que el jugador coja objetos (código en rojo):

```
int pesoTransportado;
```

Y para el tercer punto introducimos el siguiente código (la línea en rojo):

```
if (strcmp(comando,"coger") == 0)
{
	strcpy(palabra,strtok(0,"\0"));
	if (palabra == 0)
		printf("\n\nNecesito que me digas que tengo que coger\n\n");
	else
	{
	    hayObjetos = 0;
	    i = 0;
	    while (hayObjetos == 0 && i<4)
	    {
	        if (strcmp(objetos[i].nombre,palabra) == 0
	    		&& objetos[i].localizacion == habitacion+1)
	        {
	    		hayObjetos = 1;
	    		if (objetos[i].peso + pesoTransportado <= 6)
	    		{
	    			objetos[i].localizacion = -1;
	    			printf("\n\nHe cogido %s\n\n",palabra);
```

```
                                pesoTransportado += objetos[i].peso;
                        }
                        else
                                printf("\n\nNo puedo transportar mas peso\n\n");
                        i++;
                }
                if (hayObjetos == 0)
                        printf("\n\nNo puedo hacer eso\n\n");
        }
}
```

Objetos que cambian de estado

Hasta ahora muy pocas características de z88dk nuevas hemos introducido. Aprovechamos este apartado para indicar cómo añadir colores a nuestras aventuras conversacionales gracias a esta librería. Para ello introducimos un nuevo objeto, la antorcha, que es especial, porque vamos a poder encenderla y apagarla. Antes habíamos comentado que este tipo de objetos que se encienden y se apagan tienen que ser creados como dos objetos distintos. En el caso de la antorcha, podemos introducir el siguiente código en la función inicializarObjetos:

```
c[0] = 27;
c[1] = '\0';
strcpy(objetos[2].nombre,"una antorcha ");
strcat(objetos[2].nombre,c);
strcat(objetos[2].nombre,"[44mapagada");
strcat(objetos[2].nombre,c);
strcat(objetos[2].nombre,"[47m");
objetos[2].localizacion = 2;
objetos[2].peso = 2;

strcpy(objetos[3].nombre,"una antorcha ");
strcat(objetos[3].nombre,c);
strcat(objetos[3].nombre,"[43mencendida");
strcat(objetos[3].nombre,c);
strcat(objetos[3].nombre,"[47m");
objetos[3].localizacion = -2;
objetos[3].peso = 2;
```

Para que funcione tenemos que declarar la variable c al principio de la función como:

```
char c[2];
```

Se han creado dos objetos. El objeto de índice 2 se corresponde con la antorcha apagada y el objeto de índice 3 con la antorcha encendida. La antorcha apagada en un principio se encuentra en la localización con identificador 2 (el callejón) y tiene peso 2. La antorcha encendida, evidentemente, tiene el mismo peso, y al campo localizacion se le ha asignado un valor -2; vamos a usar el valor -2 en la localización para designar a aquellos objetos que no existen en el juego por alguna razón (por ejemplo, porque se hayan destruido, porque se tengan que contruir, objetos que representan diversos estados de un objeto, como por ejemplo una antorcha encendida apagada o encendida o un baúl abierto o cerrado, etc.). Esto quiere decir que si nos desplazamos a la localización 2, encontraremos la antorcha apagada, pero será imposible encontrar la antorcha encendida en ninguna localización de la aventura, y tampoco en nuestro inventario.

Es en el nombre de los objetos donde encontramos la dificultad. Si recordamos en la entrega anterior, cuando escribíamos texto con formato (subrayado, en negrita, etc.), se usaba printf de la siguiente manera:

```
printf("%c[4mTexto subrayado%c[24m",27,27);
```

Es decir, escribimos el carácter número 27, el símbolo [, y 4m indicando, por ejemplo, formato subrayado (o 24m, indicando texto sin subrayar). Como el carácter 27 es no imprimible, recurríamos a la facilidad que nos presentaba printf de escribir caracteres mediante %c. Para crear una cadena con color, el procedimiento es el mismo; sin embargo, la función strcpy no permite usar %c, así que lo que hacemos es ir concatenando, mediante el uso de strcat (que funciona igual que en el C del PC). Por ejemplo, en el caso de la antorcha apagada, deseamos que aparezca por pantalla la palabra 'apagada' con color de fondo azul. Copiamos en el nombre del objeto la cadena "una antorcha ", le concatenamos el carácter 27 (utilizando el pequeño truco visto en el código anterior), concatenamos el formato ([44m se corresponde con color de fondo azul), concatenamos la palabra "apagada", volvemos a concatenar el carácter 27 y el formato ([47m hace que el fondo vuelva a ser blanco); por lo tanto, entre ambos formatos especificados, el fondo

aparecerá de color azul. Lo mismo se ha realizado con la antorcha encendida, pero usando el color amarillo.

Por lo tanto, si en una cadena escribimos el carácter 27, el símbolo [, un código de color xx, y la m, a partir de ese momento, si xx se encuentra entre los valores de la siguiente lista, el texto tendrá el color de fondo correspondiente al texto:

```
40 - negro
41 - rojo
42 - verde
43 - amarillo
44 - azul
45 - magenta
46 - cyan
47 - blanco
```

Para el color del texto propiamente dicho, empleamos la misma táctica, pero utilizando los siguientes códigos de formato:

```
30 - negro
31 - rojo
32 - verde
33 - amarillo
34 - azul
35 - magenta
36 - cyan
37 - blanco
```

Si compilamos y cargamos el juego en el emulador, veremos que si nada más empezar nos desplazamos hacia el este, la antorcha estará allí, pero no la podremos coger. Esto se debe a que para coger un objeto teníamos que introducir el nombre exacto del mismo, y eso es imposible, porque el nombre exacto de la antorcha apagada es: "una antorcha {27}[44mapagada{27}[47m", que contiene un carácter que no se puede escribir (el 27).

Esto lo vamos a tener que solucionar añadiendo dos nuevos comandos al intérprete de comandos, "coger antorcha" y "dejar antorcha". El código se muestra a continuación:

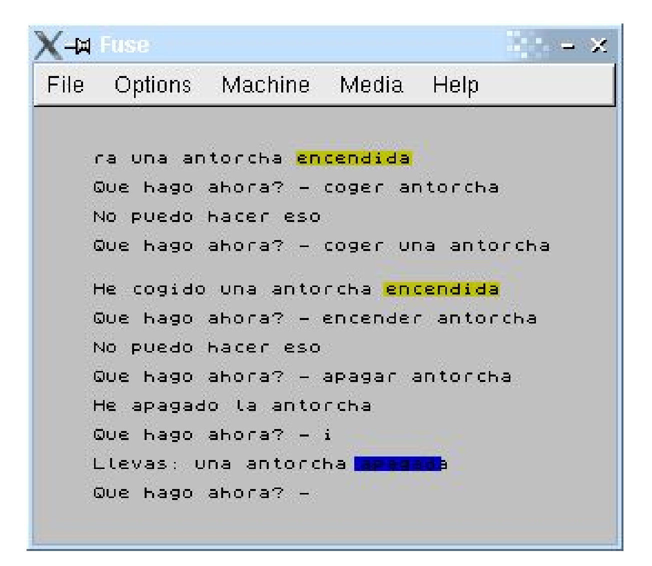
```
else if (strcmp(comando,"i") == 0 || strcmp(comando,"inventario") == 0)
{
        hayObjetos = 0;
        printf("\n\nLlevas:");
        for (i = 0; i<4;i++)
                if (objetos[i].localizacion == -1)
                {
                        printf(" %s",objetos[i].nombre);
                        hayObjetos = 1;
                }
        if (hayObjetos == 0)
                printf(" nada");
        printf("\n\n");
}
else if (strcmp(comando,"coger una antorcha") == 0)
{
        if (objetos[2].localizacion == habitacion + 1)
        {
                if (objetos[2].peso + pesoTransportado <= 6)
                {
                        objetos[2].localizacion = -1;
                        printf("\n\nHe cogido %s\n\n",objetos[2].nombre);
                        pesoTransportado += objetos[2].peso;
                }
                else
                        printf("\n\nNo puedo transportar mas peso\n\n");
        }
        else if (objetos[3].localizacion == habitacion + 1)
        {
                if (objetos[3].peso + pesoTransportado <= 6)
                {
                        objetos[3].localizacion = -1;
                        printf("\n\nHe cogido %s\n\n",objetos[3].nombre);
                        pesoTransportado += objetos[3].peso;
                }
                else
                        printf("\n\nNo puedo transportar mas peso\n\n");
        }
```

```
        else
                printf("\n\nNo puedo hacer eso\n\n");
}
else if (strcmp(comando,"dejar una antorcha") == 0)
{
        if (objetos[2].localizacion == -1)
        {
                objetos[2].localizacion = habitacion + 1;
                pesoTransportado -= objetos[2].peso;
                printf("\n\nHe dejado %s\n\n",objetos[2].nombre);
        }
        else if (objetos[3].localizacion == -1)
        {
                objetos[3].localizacion = habitacion + 1;
                pesoTransportado -= objetos[3].peso;
                printf("\n\nHe dejado %s\n\n",objetos[3].nombre);
        }
        else
                printf("\n\nNo puedo hacer eso\n\n");
}
else
// Comandos con más de una palabra
{
```

Lo que se ha hecho es aplicar el código general de coger y dejar objetos a la antorcha de tal forma que tan sólo sea necesario escribir "una antorcha" a la hora de coger o dejar el objeto, esté encendida o apagada.

Por útlimo introducimos dos comandos más, "encender antorcha" y "apagar antorcha". Para encender la antorcha es necesario que la antorcha apagada (objeto de índice 2) se encuentre en nuestro inventario (el valor de su campo localizacion debe ser -1). Lo que se hará será cambiar su valor de localizacion a -2, para que el objeto desaparezca del juego, y cambiar el de la antorcha encendida (objeto de índice 3) a -1, para que aparezca en nuestro inventario. Para apagar la antorcha se sigue el proceso contrario. Se observa que al cambiar de estado la antorcha lo único que pasa es que un objeto desaparece del juego y otro es introducido. El código se muestra a continuación:

```
}
else if (strcmp(comando,"encender antorcha") == 0)
{
        if (objetos[2].localizacion == -1)
        {
                objetos[2].localizacion = -2;
                objetos[3].localizacion = -1;
                printf("\n\nHe encendido la antorcha\n\n");
        }
        else
                printf("\n\nNo puedo hacer eso\n\n");
}
else if (strcmp(comando,"apagar antorcha") == 0)
{
        if (objetos[3].localizacion == -1)
        {
                objetos[3].localizacion = -2;
                objetos[2].localizacion = -1;
                printf("\n\nHe apagado la antorcha\n\n");
        }
        else
                printf("\n\nNo puedo hacer eso\n\n");
}
else
// Comandos con más de una palabra
{
```



La antorcha en acción

Resumen

¿Qué es lo que hemos visto en esta entrega? Resumimos:

- Cómo añadir objetos a nuestro juego, y permitir manejarlos (coger, dejar, inventario, etc.).
- Cómo añadir efectos de color a nuestro juego de texto (lo hemos visto de forma muy limitada, ya practicaremos más en posteriores entregas).
- El uso de strtok y strcat.

Está claro que si queremos permitir que el jugador realice más acciones lo único que tenemos que hacer es introducir nuevos comandos en el intérprete de comandos tal cómo se ha visto hasta ahora.

Hemos hablado de objetos normales y objetos que se pueden encender/apagar, pero en una aventura podemos encontrar otro tipo de objetos, como objetos que se pueden llevar puestos (una chaqueta, un sombrero), objetos que pueden contener a otros (un baúl, una petaca), etc. Para crear este tipo de objetos podemos jugar, como en el caso de la antorcha, con valores especiales del campo localizacion y con varios objetos para distintos estados de un mismo objeto (antorcha encendida/apagada, baúl abierto/cerrado, etc.). Se deja como ejercicio al lector implementar este tipo de objetos

Una curiosidad - violación de segmento

Por último, una curiosidad que el autor de este texto ha podido comprobar mientras realizaba este tutorial; el tema de las violaciones de segmento, con respecto a las cadenas. Por lo que se ha visto hasta ahora, no parece producirse una violación de segmento en ejecución si hay algún problema de memoria con punteros que se nos escape; simplemente se mostrarán caracteres extraños por pantalla... además, ¿cómo sería una violación de segmento en un Spectrum? ¿Similar a un RANDOMIZE USR 0?

LINKS

- Archivos fuente del ejemplo propuesto
  http://www.speccy.org/magazinezx/revistas/8/src/fuentes.tar.gz



METALBRAIN

CREA TU PROPIO SLIDESHOW

Aprovechando que me han pedido hacer un slideshow temático para la revista, he decidido dedicar el artículo de este número a examinar su código. Por falta de tiempo no he añadido ningún efecto, tan sólo he modificado ligeramente el primer efecto de líneas entrelazadas, que en FSC3 copiaba los atributos al principio y ahora deja los atributos en blanco y negro y los copia de la imagen original al final. También he realizado diversos cambios y optimizaciones, aunque seguro que siguen quedando partes mejorables.

Con este código, que podéis modificar a vuestro antojo para incluir nuevos efectos o mejoras, y vuestras propias fotos, veréis lo sencillo que resulta hacer un *slideshow* resultón.

Las fotos

El principal ingrediente de vuestro *slideshow* deben ser las fotos, para crearlas no tenéis mas que coger las fotos originales en formato .jpg y utilizar los programas BMP2SCR EXP (la versión PRO también es válida) y SevenuP (especialmente para el retoque, la conversión no funciona demasiado bien para fotos reales) para convertirlas al formato del Spectrum. No me voy a extender mas porque el asunto se saldría de lo que es la programación en ensamblador en sí misma.

Una vez que tengáis las fotos en formato .SCR, las podéis pasar a TAP con mi programa BIN2CODE, modificáis el código fuente del visor para que aparezca el número de fotos que tengáis en la etiqueta N_S y lo compiláis con Pasmo (ZMAC y AS80 también funcionan, para otros sería necesario hacer algunos pequeños cambios), lo pasáis a cinta con BIN2TAP y por último concatenáis los archivos del programa y las pantallas para obtener el *slideshow* completo.

El núcleo del programa

Esta parte del código no tiene en principio ninguna complicación, tan sólo se dedica a cargar una pantalla en la memoria, escoger un efecto para esa pantalla y aplicar dicho efecto para mostrarla. Todas las pantallas se cargan en la dirección 49512, que tiene la ventaja de que tan sólo se diferencia en un bit con 16384, de este hecho se van a aprovechar varios de los efectos. Una vez mostrada, se hace una pequeña pausa, se comprueba si quedan mas pantallas, y de ser así se incrementa el número de efecto y se vuelve al principio, reseteando dicho número de efecto si ya hemos mostrado todos los disponibles.

El código ya está comentado, así que no necesitaré extenderme demasiado. Vamos a verlo:

```
;SLIDESHOW ampliable, Version 2 by Metalbrain

N_S             EQU     18              ; Numero de pantallas
N_E             EQU     4               ; Numero de efectos
LD_BYTES        EQU     1366            ; Rutina de carga de la ROM
```

Aquí se definen etiquetas globales, que serán sustituidas por su valor numérico donde aparezcan, por lo que no ocupan memoria y no son variables.

```
                ORG     32768   ; Direccion de comienzo del codigo

                ; Nucleo principal del slide-show

                XOR     A               ;Iniciar numero de pantalla a 0
                LD      (N_SCREEN),A
EXTBUC:         XOR     A               ;Iniciar numero de efecto a 0
INTBUC:         LD      (N_EFFECT),A    ;Guardar efecto actual

                LD      DE,17           ;Cabecera, no hace falta en verdad
                LD      IX,49152        ;
                XOR     A               ;
                SCF                     ;Se puede suprimir quitando tambien
                CALL    LD_BYTES        ; las cabeceras de las pantallas

                LD      DE,6912         ;Numero de bytes
                LD      IX,49152        ;Direccion de comienzo
                LD      A,255           ;Identificador de bloque
                SCF                     ;Carry flag a 1 para cargar
                CALL    LD_BYTES        ;Llamar a la rutina de carga
```

La rutina LD_BYTES que utilizamos para cargar, carga un bloque de DE bytes que tenga el identificador especificado por A en la dirección que indica IX. Para que cargue en lugar de verificar se necesita que la bandera de acarreo esté a 1 (por eso se utiliza SCF).

```
                LD      HL,TABLE-2
                LD      A,(N_EFFECT)    ; Seleccionar efecto
                INC     A
```

```
                LD      B,A
SELEFF:         INC     HL
                INC     HL
                DJNZ    SELEFF
```

Cada efecto tiene su dirección de comienzo en una tabla. Con esto HL queda apuntando a la dirección de la rutina del siguiente efecto, contenida en dicha tabla.

```
                LD      A,(HL)          ;Introducir efecto en la direccion
                INC     HL              ; a la que se llama con CALL
                LD      (THEJUMP1),A
                LD      A,(HL)
                LD      (THEJUMP2),A

THEJUMP:        DEFB    205             ;CALL
THEJUMP1:       DEFB    0
THEJUMP2:       DEFB    0
```

La dirección de comienzo de cada efecto se introduce directamente como dato en la llamada CALL, por lo que estamos usando código automodificable. Esto no es una buena práctica de programación en procesadores más potentes (o falla o se carga la caché), pero con el Z80 no hay problema. Al llegar a THEJUMP, se hace una llamada al efecto correspondiente.

```
                  LD      B,250            ; Pausa de 5 segundos, para que de
tiempo a
PPAUSE:          HALT                       ; ver las pantallas al usar emuladores
con
                DJNZ    PPAUSE          ; carga instantanea
```

Si preferimos usar el *slideshow* en una cinta real (o desactivar la carga rápida en el emulador que usemos), podemos bajar el valor de B, o incluso omitir totalmente esta parte.

```
                LD      A,(N_SCREEN)    ;Incrementar el contador de pantallas
                INC     A
                CP      N_S

                JR      Z,THEEND        ;Si se han visto todas ya, acabar
                LD      (N_SCREEN),A

                LD      A,(N_EFFECT)    ;Incrementar contador de efecto
                INC     A
                CP      N_E
                JR      Z,EXTBUC        ;Si se han usado todos, empezar de 0
                JR      INTBUC          ;Si no, usar el siguiente
```

Esta parte del código resulta bastante sencilla, no creo que deba explicar nada más.

Y llegamos al final:

```
THEEND:         XOR     A               ;Esperar pulsacion de tecla
WAITKEY:        IN      A,(254)
                CPL
                AND     31
                JR      Z,WAITKEY
                RET                     ;Volver al BASIC
```

Si no hay ninguna tecla pulsada, el resultado de la instrucción IN serán xxx11111, mientras que si hay pulsada alguna tecla aparecerá algún 0 en los últimos 5 bits. La rutina WAITKEY es por lo tanto una forma muy común de esperar a que se pulse una tecla.

```
                ;Variables del bucle principal del slideshow

N_EFFECT:       DEFB    0
N_SCREEN:       DEFB    0

                ;Tabla de efectos

TABLE:          DEFW    EF1
```

```
                DEFW    EF2
                DEFW    EF3
                DEFW    EF4
```

En la tabla de efectos podemos ordenar los efectos como queramos e incluso poner alguno repetido para que aparezca con mas frecuencia que los demás, lo más importante es que el número de efectos que aparezcan en la tabla sea igual que el especificado en la constante N_E definida mas arriba, si fuera menor podría introducirse basura en la llamada al efecto con nefastas consecuencias.

Primer efecto: Entrelazado de líneas

Este efecto se basa en hacer un desplazamiento lateral de la pantalla, de forma que vaya entrando, pero dividiendo las líneas de forma que las pares entren por la parte de la izquierda de la pantalla, moviéndose a la derecha, y las impares entren por la derecha, avanzando hacia la izquierda. En el efecto se realizan dos bucles, el externo se repite 32 veces, una para cada carácter que avanzamos, y el interno 96 veces (192 / 2), uno por cada pareja de líneas que desplazamos.

```
        ; EF1: Efecto entrelazado de lineas, por Metalbrain
EF1_COUNT:      DEFB    0
```

Esta variable se utilizará de contador de líneas en el bucle interno del efecto.

```
EF1:            LD      A,56
                LD      HL,22528
                LD      DE,22529
                LD      (HL),A
                LD      BC,767
                LDIR                    ;Fijar los atributos
```

Con esto fijamos los atributos de la pantalla a ink=0, paper=7. Así podremos ver el desplazamiento de las líneas en todo momento.

A partir de ahora los propios comentarios del código son bastante abundantes, así que no tendré que detenerme demasiado.

```
                LD      BC,1            ;BC = numero de caracteres que entran
                                        ; de cada linea en cada iteracion.
                                        ; Varia de 1 a 32, con 32 acabamos
                                        ; de mostrar toda la pantalla y por lo
                                        ; tanto llegar a 33 significa que
                                        ; el efecto ha terminado
```

Este valor de BC va a tener varios usos. Uno de ellos es ajustar las direcciones de origen para líneas pares y de destino en las impares a su valor correcto en cada iteración. Otro es servir de contador en la instrucción LDIR que vamos a utilizar para mover los datos.

```
EF1_EXTBUC:     LD      HL,49152+32     ;HL = direccion de origen en los
                                        ; movimientos de bytes, por lo que
                                        ; se situa en la zona de memoria
                                        ; donde hemos cargado la pantalla
                                        ; que tenemos que presentar

                                        ;Las lineas pares entran hacia la
                                        ; derecha, por lo que HL al principio
                                        ; apunta 32 bytes mas alla del
                                        ; comienzo de la pantalla. Asi al
                                        ; restar BC, conseguiremos el comienzo
                                        ; del extremo derecho que tenemos que
                                        ; pintar de la linea.

                LD      DE,16384        ;DE = destino en los movimientos de
                                        ; bytes, por lo que se situa en la
                                        ; zona de memoria correspondiente a
                                        ; la pantalla que aparece en la TV
                                        ;Al principio apunta justo al comienzo
                                        ; de la pantalla, pues es donde va a
                                        ; aparecer la primera linea.
```

```
                LD      A,96            ;96 parejas de lineas quedan por
                                        ; mover

EF1_INTBUC:     LD      (EF1_COUNT),A   ;Guardar la variable
                PUSH    HL              ;Guardar HL y DE en la pila
                PUSH    DE              ; para recuperarlos tras hacer
                PUSH    HL              ; los LDIRs
                PUSH    DE
                OR      A               ;Poner el Carry Flag a 0 para hacer
                                        ; que el SBC sea como un SUB
                SBC     HL,BC           ;Origen de los bytes que vamos a mover

                LD      A,C             ;Preservar C
                LDIR                    ;Mover linea par, entrando por la
                                        ; izquierda hacia la derecha
                LD      C,A             ;Recuperar C
```

Sabemos que B=0, por lo que es mejor preservar el valor de BC de esta manera que usando las instrucciones PUSH y POP, que consumen 11 estados cada una en lugar de 4.

```
                POP     DE              ;Recuperar DE y HL
                POP     HL
                                        ;Las lineas impares entran por la
                                        ; derecha hacia la izquierda, de forma
                                        ; que tenemos que cambiar las
                                        ; posiciones relativas de DE y HL
                                        ; (aparte de hacer que bajen 1 linea
                                        ; y se coloquen en la impar)

                                        ;En este caso DE hay que colocarlo
                                        ; pasado el final de la linea y
                                        ; restarle BC para que obtenga su
                                        ; valor final

                EX      DE,HL           ;Intercambiamos DE y HL porque DE
                                        ; no permite realizar restas. En lugar
                                        ; de poner aqui esta instruccion, se
                                        ; podria haber puesto justo antes del
                                        ; SBC y operar en las siguientes
                                        ; 6 instrucciones con E y D. Lo mismo
                                        ; da una cosa que otra.

                LD      A,32            ;Sumamos 288 a HL, 32 para colocar
                ADD     A,L             ; al final de la linea y 256 para
                LD      L,A             ; bajar a la linea impar.
                LD      A,1
                ADC     A,H
                LD      H,A
```

La forma de realizar esta suma parece complicada, pero es la mejor manera de hacerla con los registros e instrucciones que el Z80 pone a nuestra disposición. En total usamos 8 bytes y 30 estados. Una suma de 16 bits ya consume de por si 2 bytes y 11 estados, a esto le sumamos otros 3 bytes y 10 estados para cargar el valor 288 en un registro, y otros 2 bytes y 22 estados para preservar y liberar un registro con las instrucciones de pila. En total ganaríamos 1 byte a costa de 13 estados.

```
                SBC     HL,BC           ;Restamos el numero de bytes a mover
                                        ; para acabar de ajustar la direccion

                EX      DE,HL           ;Y volvemos a dejar el resultado en DE

                LD      A,C             ;Preservar valor de C
                     LD        C,224              ;Sumamos 224 a HL, lo cual es
equivalente a
                  ADD     HL,BC           ;restar los 32 que nos sobran y sumar
```

```
256
                                ;para bajar una linea
                LD      C,A     ;Recuperar valor de C
                LDIR            ;Mover la linea impar
                LD      C,A     ;Recuperar numero de bytes de nuevo
```

En este caso el hecho de que 224 sea menor que 0, favorece el uso de BC para operar directamente con una operación de 16 bits, al contrario que en caso anterior.

```
                POP     DE              ;Recuperar posiciones originales
                POP     HL              ; en pantalla y en memoria

                DEC     HL              ;hacer que la linea de HL sea par
                CALL    NEXT2_EXDEHL    ;Bajar 2 lineas para DE
                CALL    NEXT2_EXDEHL    ;Y otras 2 para HL
```

No os preocupéis si no entendéis cómo la misma rutina puede operar sobre dos registros diferentes, lo veremos más adelante.

```
                INC     HL              ;recuperar el +32 que necesitamos

                LD      A,(EF1_COUNT)   ;Obtener cuenta de lineas
                DEC     A               ;Decrementar
                JR      NZ,EF1_INTBUC   ;Si no es cero, repetir el bucle
                                        ; interno para las 96 parejas de
                                        ; lineas

                HALT                    ;Pausa
                HALT

                INC     C               ;Incrementar numero de pixels a
                                        ; mostrar en la siguiente iteracion

                LD      A,C             ;Comparar con 33
                CP      33

                JR      NZ,EF1_EXTBUC   ;Continuar el bucle externo si es
                                        ; necesario

                LD      HL,49152+6144
                LD      DE,16384+6144
                LD      BC,768
                LDIR                    ;Mover los atributos
                RET                     ;Regresar a la rutina principal
```

Ahora vamos a ver la rutina que operaba sobre dos registros distintos con la misma llamada. Aquí tenemos su comienzo:

```
NEXT2_EXDEHL:   INC     D     ; Incrementa 2 lineas + intercambio de HL y DE
NEXT_EXDEHL:    EX      DE,HL ; Incrementa 1 linea + intercambio de HL y DE
```

Como podéis observar el secreto es muy sencillo: al comienzo de la rutina se intercambian los valores de HL y DE, de tal forma que la rutina actúa primero sobre HL con el valor de DE, y luego vuelve a actuar sobre HL pero esta vez sobre el valor que tenía originalmente, y el intercambio inicial además ha dejado en DE el dato procesado la primera vez. El incremento inicial también puede ponerse debajo de la orden de intercambio (pero en este caso habría que poner INC H, obviamente), pero está puesto así para que existan otros puntos de entrada en la rutina que puedan ser útiles. En un principio la rutina fue diseñada para que pudieran usarla distintos efectos en el *slideshow* FSC3, y colocada en una zona de rutinas comunes antes de los efectos, aunque al final tan sólo este efecto utiliza la rutina, y por eso lo he movido aquí esta vez. La instrucción de intercambio también podría haberse puesto fuera de la rutina, envolviendo una de las llamadas, pero en ese caso habría que repetirla, mientras que teniéndola dentro ahorramos un byte. Y mejor dejo ya de enrollarme y continuamos viendo la parte que realiza la bajada de línea:

```
NEXT_HL:        INC     H       ;Incrementa 1 linea, si pasamos de caracter
                LD      A,H     ; se queda a 0 y se produce un aumento de
                                ; tercio
                AND     7
                RET     NZ      ;Salir si no hemos pasado de caracter
```

```
                LD      A,L
                ADD     A,32    ;Pasar al siguiente caracter
                LD      L,A
                RET     C       ;Salir si se produjo un cambio de tercio
                LD      A,H
                SUB     8       ;Quitar el aumento de tercio que no se produjo
                LD      H,A
                RET
```

Para entender esta rutina, hay que recordar el formato de las direcciones de pantalla:

```
 byte alto - byte bajo
010 tt yyy - YYY XXXXX

tt -> tercio de la pantalla, de 00 a 10 (con 11 estamos con los atributos o
fuera de la pantalla)
YYY -> fila dentro del tercio (en caracteres)
yyy -> fila dentro del carácter (en pixels)
XXXXX -> columna
```

Y tenemos que tener en cuenta 3 casos:

- cuando estamos dentro de un carácter: para bajar al siguiente carácter basta con incrementar H, ya que con esto aumentamos el campo yyy, que es lo que se requiere.
- cuando pasamos dentro de un carácter al siguiente dentro del mismo tercio, hay que pasar el campo yyy de 7 a 0, y debemos incrementar el campo YYY sumando 32 a L.
- cuando pasamos de un tercio al siguiente, hay que pasar los campos yyy e YYY a 0, e incrementar tt.

Cuando la línea es par, siempre se va a dar el caso 1, y no es necesario hacer ninguna comprobación, por eso funciona el incremento del comienzo de forma que se bajan dos líneas.

Segundo efecto: Atributos en espiral

Este efecto es algo más sencillo que el anterior. Básicamente, copiamos el gráfico en la pantalla con los atributos en negro, y vamos copiando los atributos siguiendo una espiral rectangular para ir desvelando el contenido de la pantalla.

Veamos la rutina:

```
        ; EF2: Efecto atributos espiral, por Metalbrain

EF2_COUNT:      DEFB    0
EF2:            LD      DE,16384+6144+1 ;Comienzo zona atributos + 1
                LD      HL,16384+6144   ;Comienzo zona atributos
                XOR     A               ;A = 0 (color negro)
                LD      (HL),A          ;Poner color negro en el primer
                                        ; caracter de la pantalla
                LD      BC,767          ;Numero de caracteres a borrar
                LDIR                    ;Poner pantalla en negro

                LD      DE,16384
                LD      HL,49152
                LD      BC,6144
                LDIR                    ;Copiar pixels


                LD      IX,1            ;Incremento horizontal
                LD      IY,32           ;Incremento vertical
```

Comenzamos por la esquina superior izquierda, por lo que el primer incremento horizontal es positivo (vamos a la derecha) y el vertical también (hacia abajo).

```
                LD      HL,16383+6144   ;Iniciar HL = Zona de atributos-1
                LD      A,32            ;Contador inicial
                LD      (EF2_COUNT),A   ;Iniciar variable
```

```
EF2_EXTBUC:     LD      A,(EF2_COUNT)   ;Contador
                LD      B,A             ;B = numero de caracteres a desvelar

                PUSH    IX
                POP     DE              ;DE = IX = incremento horizontal

                CALL    EF2_BUC         ;Mostrar una linea horizontal de
                                        ; atributos
```
La rutina EF2_BUC que veremos más adelante funciona especificando en DE el incremento para obtener el siguiente carácter.

```
                PUSH    DE
                POP     IX              ;IX = DE
```
Al volver, en DE aparece el valor negado del que entró, de modo que lo almacenamos para que la siguiente línea se recorra en sentido contrario.

```
                LD      A,(EF2_COUNT)
                DEC     A
                LD      (EF2_COUNT),A   ;Decrementar cuenta de bytes
```
La espiral cada vez se hace más pequeña.

```
                SUB     8               ;Restar 8 para obtener el numero
                                        ; de caracteres verticales para el
                                        ; siguiente paso

                JR      Z,EF2_END       ;Si salen 0, hemos acabado
```
Aquí se detecta cuándo se produce el final de la rutina. La última línea que se recorre va a ser horizontal porque la pantalla es mas ancha que alta.

```
                LD      B,A             ;B = numero de caracteres a desvelar

                PUSH    IY
                POP     DE              ;DE = IY = incremento vertical

                CALL    EF2_BUC         ;Mostrar linea vertical

                PUSH    DE
                POP     IY              ;IY = DE

                JR      EF2_EXTBUC      ;Repetir hasta salir
```
Con esto concluye el bucle principal del efecto, veamos ahora la subrutina para copiar un número de caracteres especificado en B incrementando HL en DE bytes cada vez.

```
EF2_BUC:        ADD     HL,DE           ;Actualizar HL con el incremento
                                        ; que le corresponde

                SET     7,H             ;Hacer que HL apunte a la zona de
                                        ; memoria alta donde se almacena el
                                        ; grafico. Esto es posible hacerlo
                                        ; asi porque las direcciones de
                                        ; almacen y de pantalla solo se
                                        ; diferencian en un bit

                LD      A,(HL)          ;Tomar valor

                RES     7,H             ;Volver a apuntar a la pantalla

                LD      (HL),A          ;Escribir valor. Con esto desvelamos
                                        ; el caracter que habia oculto.

                LD      A,B             ;Hacer un HALT solo cada 8 caracteres
                AND     7               ; desvelados (hacerlo siempre
                JR      NZ,EF2_NOHALT   ; resultaria muy lento)
```

```
                HALT

EF2_NOHALT:     DJNZ    EF2_BUC         ;Mostrar todos los caracteres que se
                                        ; han pedido

                HALT                    ;Otro HALT

                LD      A,D             ;Negar el valor de DE, con lo cual
                CPL                     ; cambiamos el sentido la proxima
                LD      D,A             ; vez que avancemos
                LD      A,E
                NEG
                LD      E,A
```

La parte alta de DE basta con complementarla, pero la baja hay que negarla.

```
EF2_END:        RET                     ;Volver de la rutina EF2_BUC, o bien
                                        ; regresar a la rutina principal al
                                        ; acabar el efecto
```

Y eso es todo lo que este efecto ha dado de sí. Ya os dije que era sencillo. Seguramente hay mejores formas de hacer esto en lugar de usar los registros IX e IY, pero tenía prisa y esto fue lo primero que se me ocurrió. Si el lector lo desea, puede intentar otras ideas mejores.

Tercer efecto: Caracteres desordenados

Este efecto es bastante espectacular: sin que se borre la pantalla anterior, los caracteres de la nueva van apareciendo de forma aparentemente aleatoria, hasta que sustituyen por completo la imagen anterior. Para lograr cubrir toda la pantalla, necesitaremos una tabla donde indicar qué caracteres han sido ya sustituidos y cuáles no. La tabla no ocupará los 768 bytes de la tabla de atributos, sino tan sólo 256 correspondientes a un tercio. En cada iteración de la rutina desvelaremos un carácter de cada tercio, con una separación entre las posiciones de forma que no quede cantoso. Para escoger qué carácter va a ser el siguiente, usaremos los bytes de la ROM como guía para saber la separación entre un carácter que aparezca y el siguiente.

Vamos al lío:

```
                           ; EF3: Efecto caracteres desordenados, por Metalbrain

EF3_FILLTABLE   EQU     49152-256       ; Tabla para el efecto 3
EF3_POINTER:    DEFW    0
EF3_COUNTER:    DEFB    0
EF3:            LD      B,0             ;Contador de 0 a 0 -> 256 iteraciones
                LD      HL,EF3_FILLTABLE;Tabla de relleno, para indicar que
                                        ; caracteres hemos volcado ya y cuales
                                        ; no
                XOR     A               ;Valor inicial para rellenar la tabla
                LD      (EF3_COUNTER),A ;Iniciar contador

EF3_FILLIT:     LD      (HL),A          ;Llenar la tabla
```

Un 0 en la tabla indicará que el carácter está libre, o sea, que aún no ha sido sustituido.

```
                INC     L
                DJNZ    EF3_FILLIT

                LD      IX,(EF3_POINTER);IX apunta a la ROM, de donde se toman
                                        ; los valores de incremento que
                                        ; indican cuantas posiciones libres
                                        ; tenemos que saltar en la tabla antes
                                        ; de quedarnos con una

EF3_EXTBUC:     LD      A,(EF3_COUNTER) ;Contador en A
                LD      B,A             ;Y en B

                AND     A               ;Ver si es 0
```

```
                LD      A,(IX+0)        ;Cargar en valor de incremento

                JR      Z,EF3_NOADJUST  ;Si B vale 0, equivale a 256 y no hay
                                        ; que ajustar el valor de incremento

EF3_ADJUST:     SUB     B               ;Restar el contador al valor de
                JR      NC,EF3_ADJUST   ; incremento hasta que obtengamos un
                ADD     A,B             ; acarreo, y entonces lo volvemos a
                                        ; sumar. Con eso logramos que el
                                        ; valor de incremento sea menor que
                                        ; el contador de posiciones libres,
                                        ; para no estar dandole vueltas a la
                                        ; tabla tontamente. En otras palabras,
                                        ; hacemos A = (A MOD B), la operacion
                                        ; del modulo

EF3_NOADJUST:   LD      B,A             ;El valor de incremento ajustado lo
                                        ; ponemos en B, para el bucle con DJNZ

                XOR     A               ;En A ponemos el valor a buscar: 0
                                        ; (no copiado todavia)
EF3_PARSE:      INC     L               ;Buscamos en la tabla B valores libres
                CP      (HL)
                JR      NZ,EF3_PARSE
                DJNZ    EF3_PARSE

                INC     (HL)            ;Marcamos el valor en la tabla

                PUSH    HL              ;Guardamos HL para mas tarde

                LD      H,128+88        ;Apuntamos a la zona de atributos
                                        ; origen
```

Hay que tener en cuenta que el valor 88 en binario es 01011000, correspondiente a la zona de atributos.

```
                LD      B,3             ;3 caracteres, uno por cada tercio

                INC     IX              ;Incrementamos IX para apuntar a un
                                        ; valor de incremento diferente en la
                                        ; ROM. Este INC puede colocarse en
                                        ; cualquier otra parte del programa

EF3_OTHERTHIRD: PUSH    BC
                PUSH    HL              ;Preservar registros

                LD      E,L             ;DE=HL-32768, misma direccion pero
                LD      A,H             ; en pantalla
                SUB     128
                LD      D,A

                LD      A,(HL)          ;Copiar atributo
                LD      (DE),A

                                        ;Pasar de direccion de atributos a
                                        ; direccion de pixels en DE y HL
```

Para realizar la operación indicada en el comentario, tan sólo tenemos que modificar el valor alto del registro, ya que el valor bajo es idéntico (el campo yyy lo ponemos a 0):

```
atributo: 010 11 0tt - YYY XXXXX
gráfico:  010 tt yyy - YYY XXXXX
```

Veamos el código:

```
                                ;       A   Carry
                LD      A,D     ;010110tt      ?
                RLCA            ;10110tt0      0
                RLA             ;0110tt00      1
                RLCA            ;110tt000      0
```

Aquí lo más sencillo sería haber usado desplazamientos, ya que lo que queremos es desplazar a la izquierda a la vez que introducimos ceros por la derecha, pero por desgracia no existen instrucciones de 1 byte que permitan desplazar directamente en A de la misma forma que existen rotaciones, de forma que podemos aprovechar que conocemos el valor de los bits que van a salir para ir introduciendo esos ceros. Si os parece lioso, repasad las microfichas con las instrucciones de rotación y observad los valores.

```
                LD      H,A
```

Aprovechamos que ha salido el bit 7 a 1 para actualizar H.

```
                AND     88              ;010tt000
                LD      D,A
```

En lugar del valor 88 para el AND, nos vale cualquier valor de la forma 01x11xxx, por ejemplo 127, y en lugar del AND también podría haberse hecho SUB 128, o XOR 128, o ADD A,128. El caso es quitar el bit 7. RES 7,A también podría usarse, pero cuesta un estado más.

```
                LD      B,8             ;8 bytes por caracter

EF3_DOCHAR:     LD      A,(HL)          ;Copiar byte de pixels
                LD      (DE),A

                INC     H               ;Apuntar a la siguientes lineas
                INC     D

                DJNZ    EF3_DOCHAR
```

Aquí no nos tenemos que preocupar del cambio de carácter, con lo que el diseño de la pantalla del Spectrum se vuelve una ventaja.

```
                POP     HL              ;Recuperar HL
                POP     BC

                INC     H               ;Apuntar al siguiente tercio

                LD      A,49            ;Pequeno desplazamiento entre los
                ADD     A,L             ; caracteres que se muestran de cada
                LD      L,A             ; tercio, para que parezca mas
                                        ; aleatorio

                DJNZ    EF3_OTHERTHIRD  ;Repetir para los 3 tercios

                POP     HL              ;Recuperar direccion de la tabla

                LD      A,(EF3_COUNTER) ;Chequear la paridad del contador y
                AND     A               ; hacer un HALT solo cuando el
                JP      PE,EF3_NOHALT   ; contador tenga paridad impar. Por
                HALT                    ; termino medio, se hara una pausa
                                        ; cada 6 caracteres copiados

EF3_NOHALT:     DEC     A               ;Repetir hasta completar todos los
                LD      (EF3_COUNTER),A ; caracteres de la pantalla
                JR      NZ,EF3_EXTBUC

EF3_END:        LD      (EF3_POINTER),IX;Actualizar el puntero, asi si varias
                HALT                    ; pantallas usan este efecto, tendran
```

```
                RET                     ; patrones diferentes de aparicion
                                        ; de los caracteres
```

Con esto acaba el efecto, espero que os haya gustado tanto como a mí, porque estoy bastante orgulloso de cómo quedó y las ideas que utilicé.

Cuarto efecto: Heat up, cool down

Este efecto se basa en el que posiblemente sea el mas sencillo de los efectos de la demoscene, el primero que se enseña en los tutoriales: el fade-in/fade-out. Estos efectos consisten en variar la paleta de colores para que, sin tocar el gráfico que hay en pantalla, éste aparezca (fade-in) o se desvanezca (fade-out) poco a poco. En Spectrum no hay paleta, pero los atributos vienen a ser como una paleta de cada carácter.

Partiendo de unos atributos a 0, para hacer un fade-in basta con ir aumentando la tinta y papel de cada carácter hasta que se alcance el valor original, momento en el que no se aumenta más para ese carácter. Lo malo es que este proceso tan sólo tiene un máximo de 7 pasos, de forma que se me ocurrió otra variante dividida en dos mitades. La primera es un proceso de "calentamiento" (de ahí el "Heat up"), en la cual se hace un fade-in como el indicado para la tinta a la vez que se sube también el color del papel, pero para éste se sigue aumentando su valor hasta llegar al blanco, independientemente del valor original, y tras una pequeña pausa, comienza el proceso de "enfriamiento" ("cool down") en el que se va decrementando tan sólo el papel hasta llegar a su valor original.

Tras la larga explicación, la rutina en sí es bastante sencilla:

```
EF4:            LD      DE,16384+6144
                LD      HL,49152+6144
                LD      BC,3            ;Numero de tercios, B=0 para dar
                                        ; 256 vueltas con DJNZ
```

En otras palabras, poniendo BC a 3 estamos metiendo un 768 en CB, que es lo que se usa haciendo un DJNZ seguido de DEC C/JR NZ.

```
EF4_FLABRI:     LD      A,(HL)          ;Coger atributo
                AND     192             ;Dejar los bits de flash y bright
                LD      (DE),A          ;Fijar flash y bright
                INC     HL
                INC     DE
                DJNZ    EF4_FLABRI      ;Procesar caracteres del tercio
                DEC     C
                JR      NZ,EF4_FLABRI   ;Repetir para los 3 tercios
```

Los atributos de flash y bright se copian al principio para no tener que tratarlos después en el proceso.

```
                LD      DE,16384
                LD      HL,49152
                LD      BC,6144
                LDIR                    ;Copiar pantalla menos atributos
                HALT
                LD      E,1             ;Heat actual en registro E
EF4_HEAT_EXT:   LD      HL,49152+6144   ;Apuntar al primer atributo
                LD      BC,3            ;C=3 tercios de B=0=256 caracteres
EF4_HEAT_INT:   LD      A,(HL)          ;Cargar atributo en A
                RES     7,H             ;Pasar a coordenada de pantalla
                AND     7               ;Aislar bits de tinta
                CP      E               ;Comparar con Heat actual
                CCF                     ;Invertir acarreo. Si CF=1, es que
EF4_NOINK:      LD      A,8             ; tenemos que incrementar la tinta
                ADC     A,(HL)          ; ademas del papel
                LD      (HL),A          ;
                SET     7,H             ;Recuperar posicion original
                INC     HL              ;Avanzar posicion
                DJNZ    EF4_HEAT_INT    ;Repetir tercio
                DEC     C
                JR      NZ,EF4_HEAT_INT ;Repetir los 3 tercios
                HALT
                HALT
                HALT
                HALT                    ;Pausa
```

```
                INC     E               ;Incrementar Heat
                LD      A,E
                CP      8
                JR      NZ,EF4_HEAT_EXT ;Si es menor que 8, seguir calentando

                LD      B,12            ;Pausa antes de pasar a enfriar el
EF4_PEAKPAUSE:  HALT                    ; el papel
                DJNZ    EF4_PEAKPAUSE

                LD      E,56            ;Cool a 7 (56 = 7*8)
EF4_COOL_EXT:   LD      HL,49152+6144   ;Apuntar al primer atributo
                LD      BC,3            ;como antes, CB = 768
EF4_COOL_INT:   LD      A,(HL)          ;Tomar atributo
                RES     7,H             ;Pasar a coordenada de pantalla
                AND     56              ;Aislar bits del papel
                CP      E               ;Comprobar si papel < Cool
                JR      NC,EF4_NOPAPER  ; si no lo es, no enfriamos
                LD      A,248           ;Sumar 248 es como restar 8
                ADD     A,(HL)          ;Enfriar papel
                LD      (HL),A
EF4_NOPAPER:    SET     7,H             ;Apuntar a la pantalla virtual
                INC     HL              ;Avanzar posicion
                DJNZ    EF4_COOL_INT
                DEC     C
                JR      NZ,EF4_COOL_INT ;Repetir bucle interno
                HALT
                HALT
                HALT
                HALT                    ;Pausa
                LD      A,E
                SUB     8               ;Bajar valor de Cool
                LD      E,A
                JR      NC,EF4_COOL_EXT ;Si no es 0, seguir enfriando
                RET                     ;Salir del efecto
```

Y eso es todo.

Conclusión

Espero que lo hayáis entendido todo y seáis capaces de hacer algún que otro efecto por vuestra cuenta, no olvidéis que lo importante es practicar. Y que os divirtáis modificando los míos o haciendo vuestros propios *slideshows*. Como de costumbre, si hay algo que no veis claro (incluso después de varias miradas), os ponéis en contacto conmigo, preferiblemente a través de la lista de correo [z80asm].

Hasta la próxima.

LINKS

- Slideshow http://www.speccy.org/magazinezx/revistas/8/tap/slshow.zip
- Código fuente http://www.speccy.org/magazinezx/revistas/8/src/slidesrc.zip
- Photomatón (web de las Badasetas) http://prada.dyndns.org/photomaton/
- Freak Show City 3: http://www.matranet.net/ECSS/index3.html (enlace provisional)
- BMP2SCR: http://lcd-one.da.ru/
- SevenuP, BIN2CODE, BIN2TAP: http://www.speccy.org/metalbrain/
- Pasmo: http://www.arrakis.es/~ninsesabe/pasmo/

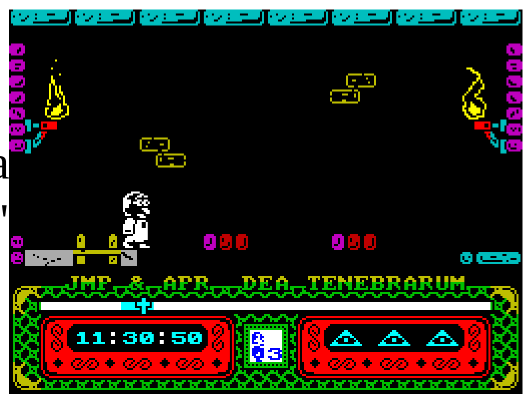


José María Pérez (de JMP & APR)

BREVE HISTORIA JMP&APR SEGÚN JMP

Para este número contamos con un testimonio de primera mano de un desarrollador de videojuegos, Jose María Pérez Rosado. Entre él y su hermano Antonio desarrollaron el juego 'Dea Tenebrarum', juego que lograron publicar comercialmente bajo el sello español System 4.

"Sólo con ver un juego, podíamos imaginar como estaba hecho por dentro."



Este artículo es una breve e interesante reseña de sus comienzos y vivencias en su etapa como desarrollador de software para ZX Spectrum.

Empezamos a hacer videojuegos a principios de los ochenta, con un Spectrum de 48k y teclado de goma, todo ello conseguido por cortesía de sus majestades los Reyes Magos, en la tele del salón, con el radiocassette de la casa. Recuerdo que la primera noche que el ordenador pasó en su nuevo hogar ya nos liamos con el tema de los videojuegos, sin tener la más mínima idea de programación, empeñados en averiguar como lograr el desplazamiento de una letra por la pantalla, cosa que finalmente conseguimos.

Y, poco a poco, tras un corto aprendizaje, en la primera semana terminé mi primer videojuego "serio", que consistía en unas pelotas que iban cayendo del cielo y que debían ser recogidas desde abajo en una especie de plato para evitar que se despachurraran, contándose en un marcador el número de pelotas buenas.

Mi siguiente juego fue realizado durante la semana siguiente, esta vez la cosa era más seria, una nave que tenía que esquivar continuos misiles que se acercaban a distintas velocidades, al mismo tiempo que cuidar la cantidad de combustible restante mediante un contador que iba disminuyendo con el tiempo y unos tanques de "fuel" que iban apareciendo a ras del suelo. Simulaba un "scroll" horizontal muy simple..

Cuando el Spectrum cumplía un mes en casa, mi hermano y yo llevamos a cabo nuestra primera colaboración. La complejidad seguía aumentando, los personajes eran bastante más grandes y numerosos. El resultado fue bautizado "Smurfy" y consistía en un hombrecillo que se movía dentro de una única pantalla por pasillos, subiendo y bajando escaleras, evitando en lo posible ser alcanzado por unas brujas montadas en escobas (brujas que eran confundidas por ciervos por algunos observadores, que su dios les conserve la vista). Técnicamente se trataba de un enorme programa, hecho en un Basic de principiantes y que, no obstante, funcionaba. Lo enviamos a Microhobby; lo rechazaron amablemente. Supongo que el motivo fue que ocupaba demasiado para lo que ofrecía.

Pero no nos desanimamos, ni mucho menos. Seguimos estudiando, aprendiendo, jugando y, en definitiva, divirtiéndonos con el ordenador. Puntualmente, cada semana, comprábamos el Microhobby para devorarlo con avidez, comentando incluso los pormenores de sus artículos, sacándoles todo el jugo. Recuerdo, por ejemplo, aquellas

entrevistas a programadores españoles, como la de Víctor Ruiz hablando de sus inicios, del Saimazoom, del Babaliba, etc. o la de Paco Suárez.

Y a los seis meses, llegó el verano, cuando decidí aprovechar todo ese tiempo libre (tenía entonces catorce años, acababa de terminar primero de bachillerato) para embarcarme en un gran proyecto en solitario. Tenía en mente un gran juego en Basic, que se concretó en un gigantesco castillo dividido en ciento veinte pantallas por las que se iba desplazando un pequeño gusano, que incluso saltaba y caía por efecto de la gravedad. La mala suerte quiso que la cinta donde lo grabé, completamente terminado ya, tras un mes de intenso trabajo, se estropeara, quedando todo mi esfuerzo perdido, sin haberme dado tiempo a enviar una copia a Microhobby. Mas poco tiempo después, cuando menos lo esperaba, un buen día la cinta funcionó. Pude recuperar el juego, lo envié a Microhobby y lo publicaron directamente en cinta. El juego se llamó WormCastle, aunque creo recordar que los de Microhobby cambiaron este altisonante nombre por el más sencillo y poco afortunado de "Gusanín", no te jode (sic). Pero publiqué mi primer juego y me embolsé quince mil pesetillas de las de aquel entonces.

Seguimos aprendiendo, por separado, mi hermano dedicado a unas cuestiones y yo a otras, pero siempre orientadas a los juegos. Hice muchos pequeños programas, incluido uno de diseño gráfico publicado en Microhobby. Dejé inacabado, aunque bastante avanzado, el mejor juego que he hecho en Basic, llamado Nosferatu. Tenía unas veinte pantallas o poco más. Incluía un ascensor que subía y bajaba suavemente, accionado a voluntad por unas palancas. Ponía a disposición del jugador numerosos objetos con utilidades concretas, necesarios para terminar la partida con éxito. Tanto el protagonista como el resto de personajes poseían movimientos más realistas y complejos que en mis juegos anteriores. Era una pequeña maravilla, pero lo dejé porque cayó en nuestras manos el primer libro de código máquina, un pequeño tesoro por el que cualquier sacrificio era poco. Era un libro de portada oscura titulado "El libro del Código Máquina del Spectrum", si no me equivoco. Más tarde llegaría el famoso curso de código máquina de Microhobby y sus utilísimas fichas de este lenguaje.

Tras algunas pruebas en solitario, escarceos, trozos de juegos, ideas dispersas por parte de ambos, nos reunimos mi hermano y yo con la firme intención de hacer un juego en código máquina, comprobados los resultados tan satisfactorios del uso de dicho lenguaje.

De continuo nos enfrentábamos a problemas aparentemente irresolubles, necesidad de técnicas que desconocíamos y no obstante existían puesto que se podían observar en algunos juegos. Así que poco a poco, tirando de imaginación, fuimos solventado todo esto y el juego fue avanzando. Su nombre era "Dea Tenebrarum", nombre que le robé a un amigo que lo ideó para un hipotético grupo de música heavy que en el futuro formaría, aunque creo que nunca lo hizo. Ya programábamos bastante bien. Sólo con ver un juego, podíamos imaginar como estaba hecho por dentro.

Y el Dea Tenebrarum iba llegando a su fin. Eran ciento veinte pantallas (otra vez) dentro de un castillo (otra vez). Nos repartimos las zonas y cada uno las diseñó a su gusto, igual que los personajes, aunque nos reuníamos continuamente para revisarlo todo, intercambiar ideas, opiniones y sugerencias. Lo cierto es que estábamos todo el santo día con el juego en la cabeza. Un día, incluso, llegue a soñar con la solución definitiva que eliminaría el más mínimo parpadeo de la pantalla. Al despertar no lo podía creer, pero aquello funcionó a las mil maravillas.

Los últimos días fueron terribles, llegando alguno a las veinte horas seguidas delante del ordenador, parando únicamente para comer. Por fortuna nuestro equipo de alta tecnología incluía ya un monitor de fósforo verde de doce pulgadas, un Joystick (un Quickshot II, si no me traiciona la memoria), una grabadora Philips, muy buena por cierto, un teclado profesional Saga (por el que tuvimos que hacer algo que nunca me perdonaré: vender nuestra fabulosa colección de tebeos de superhéroes, la cual incluía un buen número de volúmenes de aquellos de Vértice), y una pintoresca mesa de ruedas totalmente antiergonómica, donde montábamos todo el equipo para llevarlo a la terraza si hacía buen tiempo o al salón si había que hacer pruebas con la tele en color.

Con este panorama, podemos intentar imaginar como eran las pruebas del programa: primero se sentaba en el puesto de mando uno de los dos con un montón de cintas que almacenaban un puñado de código disperso en numerosos ficheros y los gráficos, se quitaba de en medio y luego se ponía el otro. Esto que se cuenta tan rápido y en una sola frase, podía llevarnos entre quince minutos y media hora, para que al ir a probar, en un sólo segundo, aparecieran unos signos extraños en pantalla y el Spectrum se autoreseteara. A revisarlo todo y vuelta a empezar. Y así una vez y otra. Si al menos hubiéramos tenido una disquetera...

Y, pese a todo, terminamos el Dea Tenebrarum, ya definitivamente. Entonces comenzó el paseo por diferentes compañías en busca de alguien que quisiera publicarlo. No era fácil, pues, aunque la técnica era correcta, la artística estaba en pañales. Erbe no quiso publicarlo y Dinamic tampoco, no obstante nos sirvió de tarjeta de presentación ante esta

última compañía, pasando a trabajar con ellos en plan "freelance" en un nuevo juego que iba a llamarse "Environment Division", nombre que se me ocurrió estudiando el lenguaje Cobol, pero no adelantemos acontecimientos.

Contactamos con System 4, una nueva compañía con necesidad de captar nuevos programadores y programas para su distribución. Tratamos con un tal Edgar Pladellorens, quien aceptó distribuir Dea Tenebrarum quizá con la esperanza de captarnos, a pesar de que le dejamos muy claro que sólo queríamos vender ese juego y que luego nos íbamos con Dinamic. El único inconveniente es que el protagonista del juego, un cura para más señas, no disparaba. Así que, no sin grandes problemas de memoria (hubo que simplificar el juego de caracteres especialmente diseñado, cambiándolo por el original del Spectrum con algún efecto), hicimos el cambio solicitado. El cura, más conocido como padre Allicrom (se sugiere leerlo al revés), ya disparaba, y si hubiera tenido que bailar sardanas pues la habría bailado, todo a gusto del consumidor.

Nos adelantaron cincuenta mil pesetas por las primeras copias que se distribuyeron, las restantes las cobraríamos por el sistema de royalties. Edgar, que parecía haber olvidado nuestro trato, el de irnos con Dinamic, pareció enfadarse cuando descubrió que de verdad nos íbamos con ellos. De manera que System 4 nos hizo el vacío. Tras algunas decenas de llamadas en las que Edgar estaba siempre "reunido" captamos la indirecta y decidimos olvidarnos del tema, a la mierda Edgar y a la mierda System 4. El poco contacto que tuvimos con Erbe, en la persona de Javier Cano, fue bastante correcto aunque al final no llegáramos a nada con ellos. Con Dinamic, igualmente, fue todo como la seda, la verdad es que se portaron bastante bien.

Y así, con nuestro primer gran juego en el mercado, nos lanzamos a la aventura del segundo. La parte técnica suponía un gran avance respecto al Dea Tenebrarum, pues ahora, en el Environment Division, íbamos a hacer scroll horizontal y vertical con bandas a distintas velocidades, sprites grandes, y animaciones más complejas. La rutina de impresión de sprites, a modo de ejemplo, construía, a su vez, una nueva rutina de impresión a medida del sprite que tuviera que imprimir en cada momento. Parece extraño, pero iba a velocidad de vértigo. Y todo ello, sprites y scroll, sin el más mínimo parpadeo ni efecto indeseable en pantalla, con desplazamientos muy suaves, pero no lentos.

Lo malo es que, poco a poco, sin darnos cuenta, lo que era una afición, si no una pasión, se había ido transformando en una carga. Fuimos perdiendo la ilusión. Por otro lado, disponíamos de menos tiempo, puesto que mi hermano (APR) se encontraba estudiando Informática en la universidad, y yo haciendo FP2 de Informática y prácticas en una empresa. Lo cierto es que empezaba a ser un poco agobiante. Y todo ello agravado porque, por si fuera poco, nos empeñábamos en hacerlo cada vez mejor, en exigirnos más.

De manera que, ya muy avanzado el Environment Division, tras una noche sin dormir pensando en como iba a decírmelo, mi hermano me anuncia que se retira. Y ahí empezó el fin. Yo no podía ni me apetecía seguir solo, se me antojaba una tarea imposible. Y, en cierto modo, aparte de apenado, sentí que me quitaba una enorme carga de encima. Lo que antes nos divertía, ahora nos agobiaba. La de mi hermano fue una sabia decisión.

Le comunicamos nuestra decisión a Dinamic, creo que a Nacho Ruiz, el cual pensaba tener algo de culpa por habernos dejado un poco olvidados, ídea un poco exagerada, pero nosotros sabíamos que no era ese el motivo. Devolvimos el Spectrum plus 3 (¡tenía disquetera y todo!) que nos habían proporcionado como adelanto por el primer juego que termináramos, y nos regalaron los disquetes (diez, que podían suponer unas cinco o siete mil pesetas de aquel entonces) donde aún se conservan las demos del Environment Division.

Luego, me embarqué yo solo en otros juegos, otra vez con mi viejo Spectrum y mis cintas, pero no llegue a concluir nada, aunque me embarqué en mil y una cosas.

Mi hermano aprobó unas oposiciones para un puesto en una empresa de telecomunicaciones de ámbito nacional, poco tiempo después tuve esa misma suerte. Hoy en día él se dedica a la Telemática y yo a la Informática, en dicha empresa, yo en Jaén y él en Madrid. Y de vez en cuando hablamos de aquella aventura que compartimos, aunque cada vez menos.

Alguna que otra vez e intentado volver a meterme en el mundillo de los videojuegos, a crear un grupo de programadores y dibujantes, pero no ha cuajado. Ya lo tengo olvidado definitivamente, pero no descarto volver a hacer algún juegecillo, eso sí, esta vez por puro divertimento, sin la más mínima pretensión. No se, quizá en mi flamante Psion Series 5mx de bolsillo recién adquirido, desde donde escribo estas líneas, me anime, algún día, después de hojear la carpeta donde guardo como oro en paño toda la documentación del Dea Tenebrarum, dibujos, ideas, el código completo escrito a mano y otras reliquias, me anime, digo, a realizar algún jueguecillo, quien sabe...



S.T.A.R.

En esta entrega, S.T.A.R. comenta desde su particular punto de vista la motivación que pueden tener los aficionados a la retrocomputación en general, y al Spectrum en particular, en emprender proyectos de coloreado de juegos antiguos.

Las cosas se hacen por:

a) Porque apetece hacerlas.
b) Porque no existían antes y se hace necesario tenerlas.
c) Porque no se tiene otra cosa que hacer.
d) Porque nada impide que se puedan hacer.

El ejemplo clásico, un tanto épico, se encuentra en una disciplina muy conocida: el alpinismo. El escalador que pretende coronar el Everest lo hace porque le apetece, porque pocos lo han hecho antes y su acción puede ser necesaria para destacar o promover asuntos particulares o sociales, lo escala porque antepone esta tarea a cualquier otra sin ninguna necesidad, y por último, se hace utilizando una famosa frase del gremio: "Escalo el Everest porque está allí".

Si bajamos a nivel del mar, a cota cero, a esa altitud donde la mayoría pasamos el día a día y cualquier nimiedad se nos puede convertir en una odisea igual o mayor que la de escalar la más alta de las montañas, tambien tendremos miriadas de ideas y proyectos por hacer que tambien estarán supeditadas bajo los cuatro puntos con los que se inaguraba este texto, tal vez con un pequeña diferencia, que hay cosas que se hacen sin que tenga ningún sentido el hacerlas, que si se razonan no se harían.

Una cosa que se hace, que se predispone y que se aplaude es la colorización de juegos de Spectrum. ¿Qué es ello? Pues coger un juego de Spectrum, plantificarlo en un emulador, cambiar la paleta por otra de 256 colores y ejecutarlo sobre el susodicho emulador, con lo que se obtiene el mismo juego pero con más colores. Que cosas ¿no? Si se hace está claro que es por una respuesta y una pregunta: se hace porque se puede y ¿por qué no hacerlo si se puede hacer?

Bien, para empezar la disertación no es lo mismo una colorización que un remake. Un remake es coger un juego y adaptarlo a las posibilidades y habilidades supuestamente más potentes que tiene la máquina destino en relación a la máquina origen de donde es nativo el juego en cuestión. En un remake no sólo se cambian los gráficos, las músicas y en el entorno, tambien se crea una programación nueva que responde al potencial y capacidad de la máquina en la que se desarrolla el remake, unas características que no por estar relacionadas con una máquina potente subministrarán una ejecución eficaz y fluida del juego. Sólo tenemos que ver cualquier juego para teléfonos móviles, no hace falta añadir mucho más.

En la colorización, en cambio, tenemos que lo que se modifica es solamente la paleta cromática, asunto que ofrece un colapso de fundamentos. Vamos a liarla:

El Spectrum es una máquina modesta, muy modesta, no está diseñada ni por casualidad para que se hagan juegos para ella, así que los diseñadores de juegos recurrían al ingenio, casi al instinto animal para programarlos. Aunque los ingredientes dispuestos en un Spectrum son pocos al menos se salva en que son compenetrables con un poco de maña y muchos apaños, pero nos centraremos más que nada en la parte gráfica ya que hablamos de colorizaciones.

En el Spectrum disponemos de una esmirriada paleta de ocho colores que se dualizan en dieciseis si los exponemos con brillo y sin brillo. Tenemos negro, azul, rojo, magenta, verde, cyan, amarillo y blanco. Vale, el negro no cuenta como color doble pero podemos ser más ingeniosos y en vez de pensar en cualquier otro color con o sin brillo tambien podemos presentarlo como un color base y su equivalente mate u oscuro, que a efectos prácticos es lo mismo pero su utilización ya se percibe con otras perspectivas ¿no es así? No es lo mismo decir que contamos con un blanco y con un gris que decir que tenemos un blanco brillante y otro normal, y eso que siguen siendo los mismos colores, dos, no nos inventamos ninguno.

Con un color con dos tonalidades (las definamos como brillantes o como oscuras es lo mismo) podemos crear aspectos lumínicos y aspectos más apagados, mostrar un gráfico que dé la sensación de estar iluminado o una sensación de que proyecte una sombra. Cosas de ingenio o de instinto animal, el lector decide.

De todas formas el Spectrum sigue siendo una máquina limitada por mucha buena intención o efectos que le pongamos, y muchos juegos sólo eran posibles si solamente se utilizaban dos colores, uno de fondo y otro para los gráficos propiamente dichos. El Spectrum funciona a base de atributos, cuadrículas de 8x8 pixels en las cuales solamente pueden inyectarse dos colores. Si estas cuadrículas se desplazaban en un scroll pixel a pixel en realidad lo hacían de forma virtual, la rejilla de pantalla sigue allí, no se mueven las cuadrículas, tan sólo son los pixels los que se desplazan para dar efecto de movimiento, así que si había un gráfico con azul y rojo y el consecutivo era magenta y verde, al desplazarse daría lugar a una casilla de la rejilla en la que coincidirían cuatro colores, inadmisible para el pobre Spectrum que sólo puede plasmar dos colores simultaneamente. O lo mismo sucedía si un personaje móvil invadía una rejilla ocupada por otros dos colores, daba lugar al típico emborrachamiento de atributos, toda una cuadrícula de 8x8 pixels mutaba de color en detrimento de la preferencia del gráfico.

El programador medio se veía obligado, por lo tanto, a diseñar su juego y sus gráficos en relación a esta restricción, lo hacía en relación a lo que le permitía el Spectrum, tanto a nivel gráfico como a nivel de recursos de CPU. Los había que pasaban de todo, si los atributos se mezclaban tanto daba pero el jugador acababa sintiendo en su estómago lo mismo que veía por los ojos, un revoltijo insoportable que era mejor no mirar. Tambien estaban los que iban a lo práctico, dos colores y aquí paz y ahí gloria, algunas veces por sentido práctico y otras, las más abundantes, porque de otra forma no se podía hacer. Y finalmente estaban los que pensaban los juegos en relación a la máquina sobre la que se hacían, diseñar y grafiquear el juego de acuerdo a lo que permitía el Spectrum.

Hacer un gráfico en dos colores (o tal vez sería más correcto decir en un solo color si no contamos el color base de fondo) es complicado, muy complicado. Con un solo color se puede combinar el fondo para sugerir sombreados, para dar profundidad, para dar brillos. Y no estamos hablando de la recurrente gama de grises, que es como se suele entender este asunto, hablamos de un solo color liso, uno.

Esta técnica (como la de saber combinar sapientemente los colores en brillo/oscuro) es la responsable de que el usuario de Spectrum perciba los juegos con agrado, la habilidad de combinatoria es la que hace que al usuario de Spectrum le gusten los juegos de Spectrum per se. Si fuera por jugabilidad, por colores, por adictividad, por diversión el usuario de Spectrum estaría en otra plataforma antes que en el Spectrum, cualquier otra máquina le pega mil patadas al Spectrum, y si es en cuestión de juegos... bueno, digamos que el Spectrum se convierte en la fea de la fiesta con la que nadie quiere bailar.

Entonces, y permítannos un breve paréntesis, ¿qué hace un usuario de Spectrum en el Spectrum? Principalmente el que tiene o ha tenido un Spectrum no ha sido más que por un hecho circunstancial, es el ordenador que se compró o que le compraron porque era el más barato o porque era el que tenía su vecino, nadie en su sano juicio elegiría un Spectrum si pudiera elegir entre un MSX, un Commodore, una Nintendo o una Atari. Pues ya que acaba siendo el poseedor de un Spectrum, bueno, algo bueno ha de tener esa maquinita, y lo bueno que tiene es que no existe ninguna otra máquina que con tan poco se haya llegado a hacer tanto y de tanta calidad.

Los juegos de Spectrum tienen ese equilibrio de elegante modestia con una brutal honestidad, son pobres pero con dignidad, sin vergüenza. Muchos son lo que han hecho juegos con unas restricciones que nadie quisiera, han utilizado al máximo sus recursos exhibiendo lo que jamás se ha conseguido en otra máquina: creatividad perfecta.

Cuando uno cuenta con una limitaciones muy marcadas el campo de acción se reduce permitiendo un ajustado margen para el acierto y para el desacierto. Si se hace una cosa mal el resultado es muy malo, si se hace una cosa bien el resultado es muy bueno. Es eso lo que sucede con los gráficos de Spectrum, se hacen con lo que hay y se mueven con lo que se puede, un equilibrio que ya imploraba Franco Batiatto en su tiempos.

Y llegamos al fin a las colorizaciones, juegos de Spectrum que se mueven como juegos de Spectrum pero que no se visualizan como juegos de Spectrum. Intentar suponer que aspecto tendría un juego de Spectrum con más colores es como suponer el comportamiento de un Seat Panda a la velocidad de un BMW. Por definición un juego de Spectrum con 256 colores no se movería como un juego nativo de Spectrum, se movería más fluidamente, la máquina contaría con otro hardware que, señores, los programadores ya habrían procurado utilizar y habrían diseñado los gráficos de otra manera que no fuera la simpleza de tener las mismas formas, tamaños y perspectivas pero con más colores, porque un gráfico en Spectrum se diseña como permite el hardware, no como desearía el diseñador.

A fin de cuentas todo este artículo es estéril, porque si recordamos el cuarto punto de la introducción, se pueden hacer colorizaciones y se puede jugar a ellas porque, cosas del libre albedrío humano, nada impide que se puedan hacer.